Eastern Economy Edition

FUNDAMENTALS OF ROBOTICS

Analysis & Control

Robert J. Schilling

Schilling
FUNDAMENTALS OF ROBOTICS
Analysis & Control

B2416929

FUNDAMENTALS OF ROBOTICS

Analysis and Control

ROBERT J. SCHILLING
Associate Professor
Clarkson University

Prentice-Hall of India Private Limited
New Delhi - 110 001
2003

This Fifth Indian Reprint—Rs. 195.00
(Original U.S. Edition—Rs. 4180.00)

FUNDAMENTALS OF ROBOTICS: Analysis and Control
by Robert J. Schilling

© 1990 by Prentice-Hall, Inc. (now known as Pearson Education Inc.), One Lake Street, Upper Saddle River, New Jersey 07458, U.S.A. All rights reserved. No part of this book may be reproduced in any form, by mimeograph or any other means, without permission in writing from the publisher.

ISBN-81-203-1047-0

Fifth Printing ... ... **July, 2003**

Published by Asoke K. Ghosh, Prentice-Hall of India Private Limited, M-97, Connaught Circus, New Delhi-110001 and Printed by Syndicate Binders, B-167, Okhla Industrial Area, Phase I, New Delhi-110020.

Dedicated to

Sandra, Stephen, Samantha, and Benjamin

Contents

PREFACE ***xv***

1 ROBOTIC MANIPULATION ***1***

1-1 Automation and Robots 1

1-2 Robot Classification 3

1-2-1 Drive Technologies, 3
1-2-2 Work-Envelope Geometries, 4
1-2-3 Motion Control Methods, 7

1-3 Applications 8

1-4 Robot Specifications 9

1-4-1 Number of Axes, 9
1-4-2 Capacity and Speed, 10
1-4-3 Reach and Stroke, 10
1-4-4 Tool Orientation, 11
1-4-5 Repeatability, Precision, and Accuracy, 13
1-4-6 Operating Environment, 16
1-4-7 An Example: Rhino XR-3, 16

1-5 Notation 18

1-6 Problems 22

References 24

2 DIRECT KINEMATICS: THE ARM EQUATION **25**

2-1 Dot and Cross Products 26

2-2 Coordinate Frames 29

2-3 Rotations 33

2-3-1 Fundamental Rotations, 33
2-3-2 Composite Rotations, 36

2-4 Homogeneous Coordinates 41

2-4-1 Homogeneous Coordinate Frames, 41
2-4-2 Translations and Rotations, 42
2-4-3 Composite Homogeneous Transformations, 45
2-4-4 Screw Transformations, 49

2-5 Link Coordinates 51

2-5-1 Kinematic Parameters, 51
2-5-2 Normal, Sliding, and Approach Vectors, 53
2-5-3 The Denavit-Hartenberg (D-H) Representation, 53
2-5-4 An Example: Microbot Alpha II, 54

2-6 The Arm Equation 57

2-6-1 The Arm Matrix, 57
2-6-2 The Arm Equation, 60
2-6-3 An Example: Microbot Alpha II, 60

2-7 A Five-Axis Articulated Robot (Rhino XR-3) 62

2-7-1 The Link-Coordinate Diagram, 63
2-7-2 The Arm Matrix, 63
2-7-3 Joint Coupling, 67

2-8 A Four-Axis SCARA Robot (Adept One) 68

2-8-1 The Link-Coordinate Diagram, 68
2-8-2 The Arm Matrix, 70

2-9 A Six-Axis Articulated Robot (Intelledex 660) 71

2-9-1 The Link-Coordinate Diagram, 73
2-9-2 The Arm Matrix, 74

2-10 Problems 76

References 80

3 INVERSE KINEMATICS: SOLVING THE ARM EQUATION **81**

3-1 The Inverse Kinematics Problem 81

3-2 General Properties of Solutions 84

3-2-1 Existence of Solutions, 84
3-2-2 Uniqueness of Solutions, 85

3-3 Tool Configuration 87

3-3-1 Tool-Configuration Vector, 87
3-3-2 Tool Configuration of a Five-Axis Articulated Robot, 88
3-3-3 Tool Configuration of a Four-Axis SCARA Robot, 89

3-4 Inverse Kinematics of a Five-Axis Articulated Robot (Rhino XR-3) 90

3-4-1 Base Joint, 92
3-4-2 Elbow Joint, 92
3-4-3 Shoulder Joint, 93
3-4-4 Tool Pitch Joint, 94
3-4-5 Tool Roll Joint, 94
3-4-6 Complete Solution, 95

3-5 Inverse Kinematics of a Four-Axis SCARA Robot (Adept One) 96

3-5-1 Elbow Joint, 97
3-5-2 Base Joint, 98
3-5-3 Vertical Extension Joint, 98
3-5-4 Tool Roll Joint, 98
3-5-5 Complete Solution, 99

3-6 Inverse Kinematics of a Six-Axis Articulated Robot (Intelledex 660) 100

3-6-1 Tool Roll Joint, 101
3-6-2 Shoulder Roll Joint, 101
3-6-3 Base Joint, 102
3-6-4 Elbow Joint, 102
3-6-5 Shoulder Pitch Joint, 103
3-6-6 Tool Pitch Joint, 104
3-6-7 Complete Solution, 104

3-7 Inverse Kinematics of a Three-Axis Planar Articulated Robot 105

3-7-1 Shoulder Joint, 107
3-7-2 Base Joint, 108
3-7-3 Tool Roll Joint, 108
3-7-4 Complete Solution, 108

3-8 A Robotic Work Cell 109

3-9 Problems 112

References 115

4 WORKSPACE ANALYSIS AND TRAJECTORY PLANNING 116

4-1 Workspace Analysis 116

4-2 Work Envelope of a Five-Axis Articulated Robot (Rhino XR-3) 118

4-3 Work Envelope of a Four-Axis SCARA Robot (Adept One) 122

4-4 Workspace Fixtures 124

4-4-1 Part Feeders, 125
4-4-2 Conveyers and Carousels, 127
4-4-3 Fixed Tools, 128

4-5 The Pick-and-Place Operation 131

4-5-1 Pick and Lift-Off Points, 131
4-5-2 Place and Set-Down Points, 132
4-5-3 Speed Variation, 134

4-6 Continuous-Path Motion 135

4-6-1 Paths and Trajectories, 135
4-6-2 Continuous-Path Control of a Five-Axis Articulated Robot (Rhino XR-3), 137
4-6-3 Continuous-Path Control of a Four-Axis SCARA Robot (Adept One), 139

4-7 Interpolated Motion 140

4-7-1 Cubic Polynomial Paths, 141
4-7-2 Linear Interpolation with Parabolic Blends, 142

4-8 Straight-Line Motion 145

4-9 Problems 148

References 151

5 DIFFERENTIAL MOTION AND STATICS **153**

5-1 The Tool-Configuration Jacobian Matrix 153

5-1-1 Tool Jacobian Matrix of a Five-Axis Articulated Robot (Rhino XR-3), 154
5-1-2 Tool Jacobian Matrix of a Four-Axis SCARA Robot (Adept One), 156
5-1-3 Tool Jacobian Matrix of a Three-Axis Planar Articulated Robot, 157

5-2 Joint-Space Singularities 158

5-3 Generalized Inverses 160

5-4 Resolved-Motion Rate Control: $n \leq 6$ 164

5-5 Rate Control of Redundant Robots: $n > 6$ 166

5-6 Rate Control Using {1}-Inverses 171

5-7 The Manipulator Jacobian 174

5-7-1 Manipulator Jacobian of a Four-Axis SCARA Robot (Adept One), 177
5-7-2 Manipulator Jacobian of a Five-Axis Articulated Robot (Rhino XR-3), 179
5-7-3 Manipulator Jacobian of a Three-Axis Planar Articulated Robot, 181

5-8 Induced Joint Torques and Forces 182

5-8-1 End-of-Arm Compliance and Stiffness, 184
5-8-2 Joint-Space Singularities, 187

5-9 Problems 190

References 193

6 MANIPULATOR DYNAMICS **194**

6-1 Lagrange's Equation 195
6-2 Kinetic and Potential Energy 195

6-2-1 Link Inertia Tensor, 196
6-2-2 Link Jacobian, 198
6-2-3 Manipulator Inertia Tensor, 199
6-2-4 Gravity, 200

6-3 Generalized Force 201

6-3-1 Actuators, 202
6-3-2 Friction, 202

6-4 Lagrange-Euler Dynamic Model 204

6-5 Dynamic Model of a Two-Axis Planar Articulated Robot 208

6-6 Dynamic Model of a Three-Axis SCARA Robot 212

6-7 Direct and Inverse Dynamics 220

6-8 Recursive Newton-Euler Formulation 221

6-8-1 Forward Newton-Euler Equations, 222
6-8-2 Backward Newton-Euler Equations, 223

6-9 Dynamic Model of a One-Axis Robot (Inverted Pendulum) 226

6-9-1 Lagrange-Euler Formulation, 226
6-9-2 Newton-Euler Formulation, 228

6-10 Problems 231

References 232

7 ROBOT CONTROL **234**

7-1 The Control Problem 235

7-2 State Equations 236

7-2-1 A One-Axis Robot (Inverted Pendulum), 237
7-2-2 A Two-Axis Planar Articulated Robot, 238
7-2-3 A Three-Axis SCARA Robot, 240

7-3 Constant Solutions 243

7-3-1 Liapunov's First Method, 247
7-3-2 Liapunov's Second Method, 250

7-4 Linear Feedback Systems 256

7-4-1 Transfer Function, 256
7-4-2 Steady-State Tracking, 261
7-4-3 Transient Performance, 263

7-5 Single-Axis PID Control 265

7-5-1 DC Motor and Load, 266
7-5-2 Torque Regulator, 268
7-5-3 PID Transfer Function, 272

7-6 PD-Gravity Control 276

7-7 Computed-Torque Control 283

7-8 Variable-Structure Control 289

7-9 Impedance Control 298

7-10 Problems 303

References 306

8 ROBOT VISION **307**

8-1 Image Representation 308

8-2 Template Matching 309

8-3 Polyhedral Objects 313

8-3-1 Edge Detection, 313
8-3-2 Corner Points, 317
8-3-3 Run-Length Encoding, 318

8-4 Shape Analysis 319

8-4-1 Line Descriptors, 319
8-4-2 Area Descriptors, 321
8-4-3 Principal Angle, 324

8-5 Segmentation 325

8-5-1 Thresholding, 325
8-5-2 Region Labeling, 326

8-6 Iterative Processing 330

8-6-1 Shrink Operators, 330
8-6-2 Swell Operators, 332
8-6-3 Euler Number, 334

8-7 Perspective Transformations 337

8-7-1 Perspective Transformation, 337
8-7-2 Inverse Perspective Transformation, 340
8-7-3 Pixel Coordinates, 343

8-8 Structured Illumination 345

8-8-1 Light Sources, 345
8-8-2 Light Patterns, 346
8-8-3 Triangulation, 348

8-9 Camera Calibration 350

8-10 Problems 353

References 355

9 TASK PLANNING **357**

9-1 Task-Level Programming 358

9-2 Uncertainty 359

9-3 Configuration Space 362

9-3-1 Translations, 362
9-3-2 Rotations, 366

9-4 Gross-Motion Planning 368

9-4-1 Generalized Voronoi Diagrams (GVD), 369
9-4-2 Motion Heuristics, 375

9-5 Grasp Planning 378

9-6 Fine-Motion Planning 381

9-6-1 Guarded Motion, 382
9-6-2 Compliant Motion, 385

9-7 Simulation of Planar Motion 388

9-8 A Task-Planning Problem 391

9-8-1 Source and Goal Scenes, 391

9-8-2 Task-Planning Subproblems, 392
9-8-3 Task Planner Simulation, 394

9-9 Problems 396

References 398

APPENDIX 1 TRIGONOMETRIC IDENTITIES ***400***

APPENDIX 2 MOMENTS OF INERTIA ***402***

APPENDIX 3 LIST OF SYMBOLS ***405***

INDEX ***413***

Preface

Robotics is an exciting, dynamic interdisciplinary field of study. The purpose of this book is to introduce the reader to the *fundamentals* of robotics and to the analysis and control of industrial robots. This book is based on class notes used in a two-course sequence taught at Clarkson University. It is primarily intended for senior-level undergraduates and beginning graduates in the fields of electrical engineering, mechanical engineering, and computer science. It should also be useful to practicing engineers as a reference text.

The ideal background for a person setting out to study robotics includes a rather wide assortment of topics from engineering and computer science. Since few individuals have this background, a careful attempt has been made to develop the topics in such a way as to minimize the prerequisites. It is assumed that the reader has been exposed to the standard freshman and sophomore courses in calculus and elementary differential equations. Some familiarity with vectors and matrices is recommended, and for some of the material in the second half of the book, knowledge of signals and linear systems is helpful. Laboratory projects are developed in the separate Laboratory Manual that accompanies this text, and for these exercises it is assumed that the reader has programming experience with at least one high-level language.

The topics are divided into nine chapters as shown in Fig. P-1, where the arrows indicate the order in which the chapters can be covered. Generally speaking, the treatment becomes more sophisticated as one progresses to lower levels of Fig. P-1. Chapters that appear on the same level can be covered in any order.

Chapter 1 is a descriptive introduction that discusses the characterization, classification, and specification of robotic manipulators. It concludes with a brief discussion of the important role played by notation. Chapter 2, on direct kinematics,

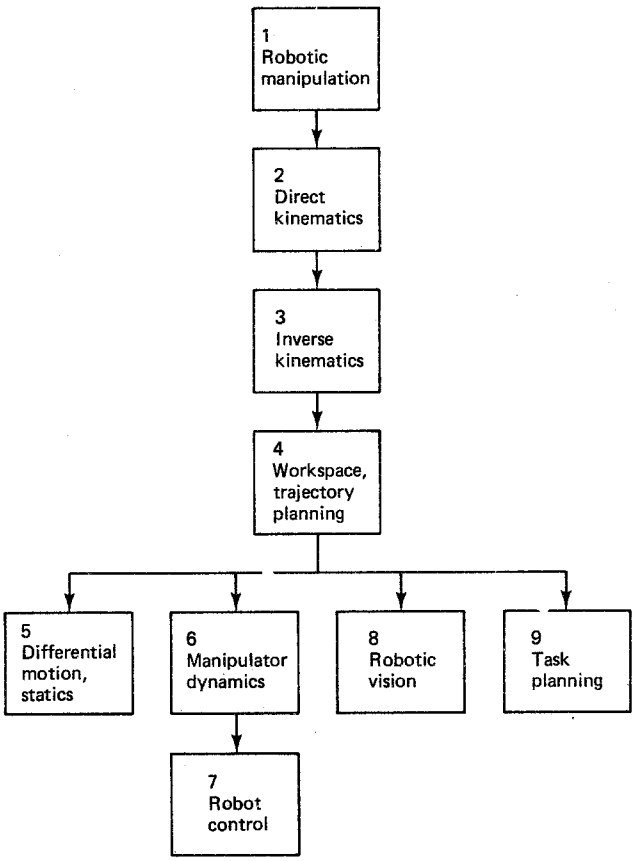


Figure P-1 Robotics topics.

is an important chapter because it lays the mathematical foundation for much of what follows. In Chap. 2, the algebraic "arm equation" relating joint angles to hand position and orientation is developed, and complete generic solutions are worked out for several important classes of robots. Chapter 3, on inverse kinematics, complements Chap. 2 by developing a solution to the arm equation. This allows the user to determine the joint angles needed to obtain desired positions and orientations of the hand. The solution of the arm equation lays a foundation for the effective use of external sensors, particularly robot vision. The last chapter in the initial progression, Chap. 4, covers workspace analysis and trajectory planning. Here the reader begins to examine techniques for planning robot motions in order to perform meaningful manipulation tasks such as pick-and-place operations.

The initial sequence of four chapters is followed by more advanced material in chapters that do not have to be taken in strict numerical order. The three chapters represented in the lower left of Fig. P-1, Chaps. 5, 6, and 7, focus on robot statics, dynamics, and control. Some familiarity with differential equations, the Laplace transform, and transfer functions is helpful here, particularly for Chap. 7. Chapter 5 covers two mathematically related topics, differential motion and statics. Differential

motion techniques provide for a simple, yet effective, way of controlling robots through speed regulation. The static analysis is useful for analyzing robots when they are in contact with their environment. Chapter 6, on manipulator dynamics, is an introduction to dynamic modeling based on the Lagrange-Euler and recursive Newton-Euler techniques. Since this material can become unwieldy, the focus is on developing equations of motion for simple one-, two-, and three-axis robots. Chapter 7, on robot control, is the most sophisticated chapter mathematically. It uses the robot models developed in Chap. 6 to illustrate a variety of torque-based control techniques including single-axis PID control, PD-plus-gravity control, computed-torque control, variable-structure control, and impedance control.

The two chapters represented in the lower right side of Fig. P-1, Chaps. 8 and 9, focus on topics which have an artificial intelligence (AI) flavor. Like the first four chapters, these chapters are algebraic in nature; they do not require any knowledge of differential equations. Chapter 8 is an introduction to the fundamentals of robot vision. It focuses on practical analysis techniques applicable to binary images obtained from a stationary overhead camera. The overall objective is to recognize objects in the workspace and determine their positions and orientations so they can be successfully manipulated by the robot. Chapter 9, on task planning, outlines a high-level approach to robot programming. Task-planning techniques are based on specification of manipulation goals, rather than specific means used to achieve these goals. The function of the task planner is to synthesize a series of detailed instructions for the robot controller in order to achieve a high-level manipulation task.

Numerous examples and exercises are included throughout the book. The examples illustrate important special cases of results developed in the text, while the exercises serve to test the reader's understanding of the material. There is a generous set of problems at the end of each chapter; complete solutions to these problems are available in the accompanying Instructor's Manual. A careful selection of industrial, educational, and generic robots have been used throughout the text and in the problems. They range from simple one-, two-, and three-axis robots used to investigate dynamics and control to more complex four-, five-, six-, and n-axis robots used to examine kinematics, workspace analysis, trajectory planning, and differential motion. Several specific robotic manipulators are employed as case study type examples throughout the text as summarized in Table P-1.

TABLE P-1 EXAMPLE ROBOTS

Robot	Axes	Geometry	Type
Planar	3	Articulated	Generic
Adept One	4	SCARA	Industrial
Rhino XR-3	5	Articulated	Educational
Intelledex 660	6	Articulated	Industrial

The three-axis planar articulated robot is a general planar manipulator and is the simplest of the four robots listed. The four-axis Adept One robot is a commercial direct-drive SCARA-type robotic arm that is useful for high-speed assembly operations. It is a kinematically simple manipulator, yet it is an eminently practical one

as well. The five-axis Rhino XR-3 educational robot is an articulated arm designed to be a vehicle for teaching robotics. It has an open design with features similar to those of the larger industrial robots, yet it is less expensive and it is safer to use in an instructional setting. The six-axis articulated robot is a sophisticated general-purpose commercial robot that is designed for light assembly operations and for clean room material handling.

This book is accompanied by a Laboratory Manual, which contains a carefully integrated series of exercises that closely follow the chapters. These programming projects serve both as a motivational vehicle and as a reinforcement mechanism for the lecture material. Use of the Laboratory Manual is optional, but highly recommended, because experience shows that students are more *motivated* to follow theoretical developments if they see the underlying principles implemented in a laboratory setting. Robotics is an ideal medium for this type of interaction because it can provide direct hands-on feedback to the student.

The laboratory exercises are designed around a software package that features a three-dimensional graphics robot simulator that can be used to develop and test robot control programs off-line on an IBM PC/XT/AT or compatible computer using a variety of programming languages. With the use of the simulator, it is possible to include laboratory projects in large robotics classes and to teach *complete* robotics courses even when instructional robots are not physically available. The laboratory exercises are designed around an inexpensive robotic work cell that features an educational robot, the Rhino XR-3, and an overhead vision system, the Micron Eye. Instructional software is supplied which includes source code solutions to the laboratory exercises and supplementary programs useful for classroom demonstrations.

This book and its accompanying Laboratory Manual were designed for a two-semester course in robotics taught at the senior undergraduate and beginning graduate level. The book can be effectively used for shorter courses if selected material is omitted. Suggested chapter sequences are summarized in Table P-2, where the letter L stands for laboratory projects. If the laboratory projects are not included, or if the course is taught to more advanced students, then additional material can be covered.

TABLE P-2 SUGGESTED CHAPTER SEQUENCES

Course Duration	Chapters
1 quarter	1, 2, 3, 4, L
1 semester	1, 2, 3, 4, 5, L or 1, 2, 3, 4, 8, L
2 quarters	1, 2, 3, 4, L 5, 6, 7, L } or { 1, 2, 3, 4, L 5, 8, 9, L
2 semesters	1, 2, 3, 4, 5, L 6, 7, 8, 9, L
3 quarters	1, 2, 3, 4, L 5, 6, 7, L 7, 8, 9, L

ACKNOWLEDGMENTS

Many individuals have contributed to this book. For providing the necessary facilities and the proper atmosphere within which to work, I wish to acknowledge Clarkson University. The Westinghouse Educational Foundation provided direct support for this effort, and for this I am grateful. I am also indebted to the students who read early drafts of the manuscript and contributed to its revisions through their questions, comments, and suggestions. This list includes, in particular, N. ElSibai, S. Fournier, A. Fuchs, T. Hemminger, E. Ibrahim, B. Lynch, K. May, A. Pradeep, J. Tang, and S. Williams. Special thanks to M. Sivakumar, who prepared the Instructor's Manual; O. Takahashi, who did the gross-motion path-planning simulations; R. White, who developed the graphics robot simulator; and J. Wu, who did the control simulations. A number of my colleagues offered to review preliminary drafts of the manuscript, and for this assistance my thanks go to M. Manoochehri, R. Mukundan, R. Read, G. Walker, and P. Yoder. The anonymous reviewers at Prentice Hall also made a number of valuable suggestions. Finally, I wish to acknowledge the patience and support of my wife, Sandra, and my children, who allowed me to slip away into the study for prolonged periods of time in order to complete the manuscript.

R. Schilling
Potsdam, New York

FUNDAMENTALS OF ROBOTICS
Analysis and Control

1

Robotic Manipulation

The term *robot* can convey many different meanings in the mind of the reader, depending on the context. In the treatment presented here, a robot will be taken to mean an industrial robot, also called a *robotic manipulator* or a *robotic arm*. An example of an industrial robot is shown in Fig. 1-1. This is an articulated robotic arm and is roughly similar to a human arm. It can be modeled as a chain of rigid links interconnected by flexible joints. The links correspond to such features of the human anatomy as the chest, upper arm, and forearm, while the joints correspond to the shoulder, elbow, and wrist. At the end of a robotic arm is an end-effector, also called a *tool, gripper,* or *hand*. The tool often has two or more *fingers* that open and close.

To further characterize industrial robots, we begin by examining the role they play in automation in general. This is followed by a discussion of robot classifications using a number of criteria including: drive technologies, work envelope geometries, and motion control methods. A brief summary of the most common applications of robots is then presented; this is followed by an examination of robot design specifications. Chap. 1 concludes with a discussion of the use of notation and a summary of the notational conventions adopted in the remainder of the text.

1-1 AUTOMATION AND ROBOTS

Mass-production assembly lines were first introduced at the beginning of the twentieth century (1905) by the Ford Motor Company. Over the ensuing decades, specialized machines have been designed and developed for high-volume production of mechanical and electrical parts. However, when each yearly production cycle ends and new models of the parts are to be introduced, the specialized machines have to be shut down and the hardware *retooled* for the next generation of models. Since periodic modification of the production *hardware* is required, this type of automation is

referred to as *hard automation*. Here the machines and processes are often very efficient, but they have limited flexibility.

More recently, the auto industry and other industries have introduced more flexible forms of automation in the manufacturing cycle. *Programmable mechanical manipulators* are now being used to perform such tasks as spot welding, spray painting, material handling, and component assembly. Since computer-controlled mechanical manipulators can be easily converted through *software* to do a variety of

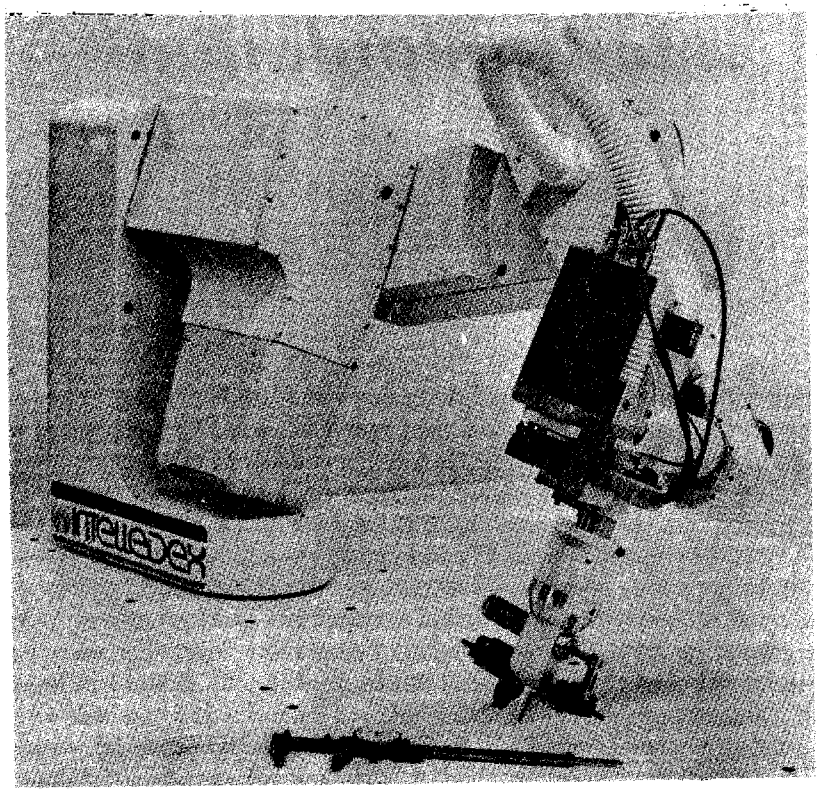


Figure 1-1 An industrial robot. (Courtesy of Intelledex, Inc., Corvallis, OR.)

tasks, they are referred to as examples of *soft automation*. A qualitative comparison of the cost effectiveness of manual labor, hard automation, and soft automation as a function of the production volume (Dorf, 1983) is summarized in Fig. 1-2.

It is evident that for very low production volumes, such as those occurring in small batch processing, manual labor is the most cost-effective. As the production volume increases, there comes a point v_1 where robots become more cost-effective than manual labor. As the production volume increases still further, it eventually reaches a point v_2 where hard automation surpasses both manual labor and robots in cost-effectiveness. The curves in Fig. 1-2 are representative of general qualitative trends, with the exact data dependent upon the characteristics of the unit being produced. As robots become more sophisticated and less expensive, the range of pro-

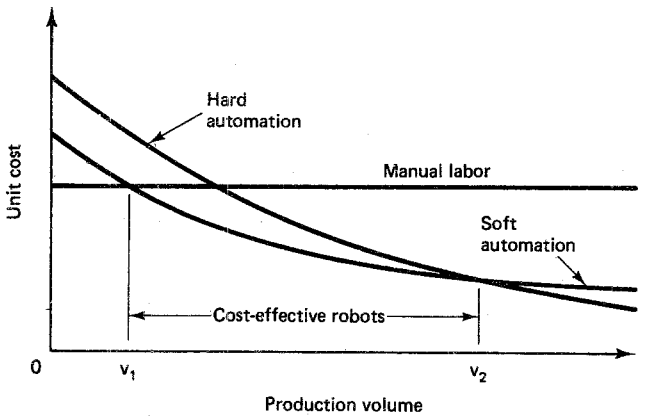


Figure 1-2 Relative cost-effectiveness of soft automation.

duction volumes [v_1, v_2] over which they are cost-effective continues to expand at both ends of the production spectrum.

In order to more clearly distinguish soft automation from hard automation, it is useful to introduce a specific definition of a robot. A number of definitions have been proposed over the years. However, as robotic technology continues to evolve, any definition proposed may need to be refined and updated before long. For the purpose of the material presented in this text, the following definition is used:

Definition 1-1-1: Robot. A *robot* is a software-controllable mechanical device that uses sensors to guide one or more end-effectors through programmed motions in a workspace in order to manipulate physical objects.

Contrary to popular notions about robots in the science fiction literature (see, for instance, Asimov, 1950), today's industrial robots are not *androids* built to impersonate humans. Indeed, most are not even capable of self-locomotion. However, many of today's robots are anthropomorphic in the sense that they are patterned after the human arm. Consequently, industrial robots are often referred to as *robotic arms* or, more generally, as *robotic manipulators*.

1-2 ROBOT CLASSIFICATION

In order to refine the general notion of a robotic manipulator, it is helpful to classify manipulators according to various criteria such as drive technologies, work envelope geometries, and motion control methods.

1-2-1 Drive Technologies

One of the most fundamental classification schemes is based upon the source of power used to drive the joints of the robot. The two most popular drive technologies are *electric* and *hydraulic*. Most robotic manipulators today use electric drives in the form of either DC servomotors or DC stepper motors. However, when high-speed

manipulation of substantial loads is required, such as in molten steel handling or auto body part handling, hydraulic-drive robots are preferred. One serious drawback of hydraulic-drive robots lies in their lack of cleanliness, a characteristic that is important for many assembly applications.

Both electric-drive robots and hydraulic-drive robots often use *pneumatic tools* or end-effectors, particularly when the only gripping action required is a simple open-close type of operation. An important characteristic of air-activated tools is that they exhibit built-in *compliance* in grasping objects, since air is a compressible fluid. This is in contrast to sensorless rigid mechanical grippers, which can easily damage a delicate object by squeezing too hard.

1-2-2 Work-Envelope Geometries

The end-effector, or tool, of a robotic manipulator is typically mounted on a flange or plate secured to the wrist of the robot. The *gross work envelope* of a robot is defined as the locus of points in three-dimensional space that can be reached by the wrist. We will refer to the axes of the first three joints of a robot as the *major* axes. Roughly speaking, it is the major axes that are used to determine the position of the wrist. The axes of the remaining joints, the *minor* axes, are used to establish the orientation of the tool. As a consequence, the geometry of the work envelope is determined by the sequence of joints used for the first three axes. Six types of robot joints are possible (Fu et al., 1987). However, only two basic types are commonly used in industrial robots, and they are listed in Table 1-1.

TABLE 1-1 TYPES OF ROBOT JOINTS

Type	Notation	Symbol	Description
Revolute	R		Rotary motion *about* an axis
Prismatic	P		Linear motion *along* an axis

Revolute joints (R) exhibit rotary motion about an axis. They are the most common type of joint. The next most common type is a *prismatic* joint (P), which exhibits sliding or linear motion along an axis. The particular combination of revolute and prismatic joints for the three major axes determines the geometry of the work envelope, as summarized in Table 1-2. The list in Table 1-2 is not exhaustive, since there are many possibilities, but it is representative of the vast majority of commercially available robots. As far as analysis of the motion of the arm is concerned, prismatic joints tend to be simpler than revolute joints. Therefore the last column in Table 1-2, which specifies the total number of revolute joints for the three major axes, is a rough indication of the complexity of the arm.

For the simplest robot listed in Table 1-2, the three major axes are all prismatic; the resulting notation for this configuration is PPP. This is characteristic of a *Cartesian-coordinate robot*, also called a *rectangular-coordinate robot*. An example

TABLE 1-2 ROBOT WORK ENVELOPES BASED ON MAJOR AXES

Robot	Axis 1	Axis 2	Axis 3	Total revolute
Cartesian	P	P	P	0
Cylindrical	R	P	P	1
Spherical	R	R	P	2
SCARA	R	R	P	2
Articulated	R	R	R	3

P = prismatic, R = revolute.

of a Cartesian-coordinate robot is shown in Fig. 1-3. Note that the three sliding joints correspond to moving the wrist up and down, in and out, and back and forth. It is evident that the work envelope or work volume that this configuration generates is a rectangular box. When a Cartesian-coordinate robot is mounted from above in a rectangular frame, it is referred to as a *gantry* robot.

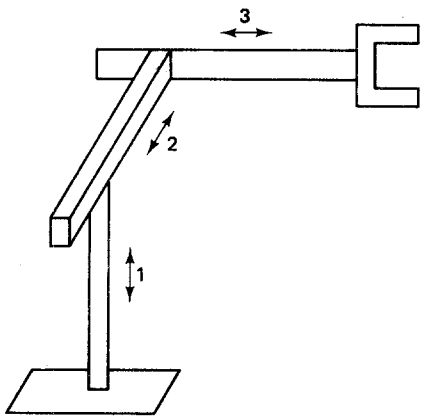


Figure 1-3 Cartesian robot.

If the first joint of a Cartesian-coordinate robot is replaced with a revolute joint (to form the configuration RPP), this produces a *cylindrical-coordinate robot*. An example of a cylindrical-coordinate robot is shown in Fig. 1-4. The revolute joint swings the arm back and forth about a vertical base axis. The prismatic joints then move the wrist up and down along the vertical axis and in and out along a radial axis. Since there will be some minimum radial position, the work envelope generated by this joint configuration is the volume between two vertical concentric cylinders.

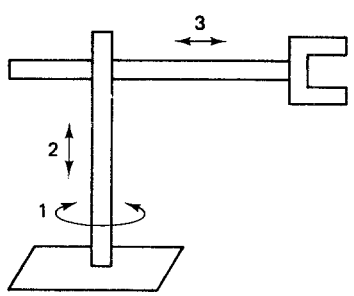


Figure 1-4 Cylindrical robot.

If the second joint of a cylindrical-coordinate robot is replaced with a revolute joint (so that the configuration is then RRP), this produces a *spherical-coordinate robot*. An example of a spherical-coordinate robot is shown in Fig. 1-5. Here the first revolute joint swings the arm back and forth about a vertical base axis, while the second revolute joint pitches the arm up and down about a horizontal shoulder axis. The prismatic joint moves the wrist radially in and out. The work envelope generated in this case is the volume between two concentric spheres. The spheres are typically truncated from above, below, and behind by limits on the ranges of travel of the joints.

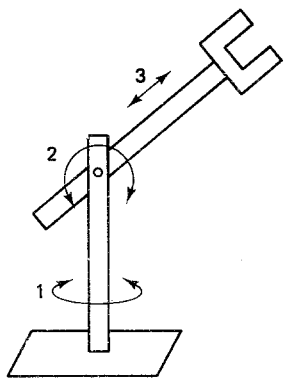


Figure 1-5 Spherical robot.

Like a spherical-coordinate robot, a *SCARA robot* (*S*elective *C*ompliance *A*ssembly *R*obot *A*rm) also has two revolute joints and one prismatic joint (in the configuration RRP) to position the wrist. However, for a SCARA robot the axes of all three joints are vertical, as shown in Fig. 1-6. The first revolute joint swings the arm back and forth about a base axis that can also be thought of as a vertical shoulder axis. The second revolute joint swings the forearm back and forth about a vertical elbow axis. Thus the two revolute joints control motion in a horizontal plane. The vertical component of the motion is provided by the third joint, a prismatic joint which slides the wrist up and down. The shape of a horizontal cross section of the work envelope of a SCARA robot can be quite complex, depending upon the limits on the ranges of travel for the first two axes.

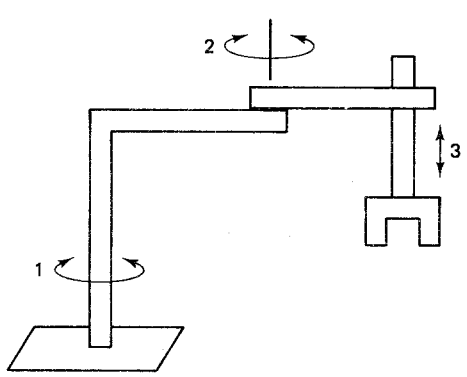


Figure 1-6 SCARA robot.

When the last remaining prismatic joint is replaced by a revolute joint (to yield the configuration RRR), this produces an *articulated-coordinate robot*. An articulated-coordinate robot is the dual of a Cartesian robot in the sense that all three of the major axes are revolute rather than prismatic. The articulated-coordinate robot is the most anthropomorphic configuration; that is, it most closely resembles the anatomy of the human arm. Articulated robots are also called *revolute robots*. An example of an articulated-coordinate robot is shown in Fig. 1-7. Here the first revolute joint swings the robot back and forth about a vertical base axis. The second joint pitches the arm up and down about a horizontal shoulder axis, and the third joint pitches the forearm up and down about a horizontal elbow axis. These motions create a complex work envelope, with a side-view cross section typically being crescent-shaped.

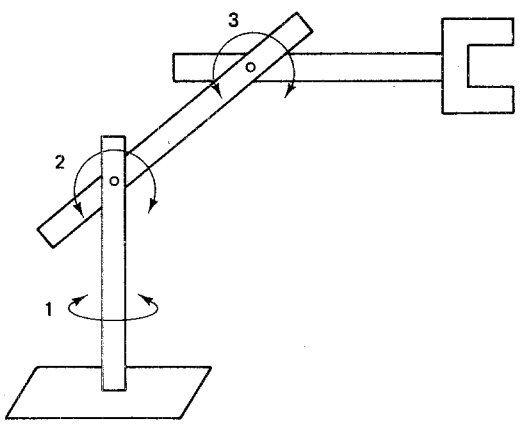


Figure 1-7 Articulated robot.

1-2-3 Motion Control Methods

Another fundamental classification criterion is the method used to control the movement of the end-effector or tool. The two basic types of movement are listed in Table 1-3. The first type is *point-to-point* motion, where the tool moves to a sequence of discrete points in the workspace. The path between the points is not explicitly controlled by the user. Point-to-point motion is useful for operations which are discrete in nature. For example, spot welding is an application for which point-to-point motion of the tool is all that is required.

TABLE 1-3 TYPES OF ROBOT MOTION CONTROL

Control method	Applications
Point to point	Spot welding Pick-and-place Loading and unloading
Continuous path	Spray painting Arc welding Gluing

The other type of motion is *continuous-path* motion, sometimes called *controlled-path* motion. Here the end-effector must follow a prescribed path in three-dimensional space, and the speed of motion along the path may vary. This clearly presents a more challenging control problem. Examples of applications for robots with continuous-path motion control include paint spraying, arc welding, and the application of glue or sealant.

1-3 APPLICATIONS

Robotic applications often involve simple, tedious, repetitive tasks such as the loading and unloading of machines. They also include tasks that must be performed in harsh or unhealthy environments such as spray painting and the handling of toxic materials. A summary of robot applications (Brody, 1987) is displayed in Table 1-4. Here the percentages listed in the second column represent shares of the overall robot market in the United States for the year 1986. The size of the market in 1986, excluding separate sales of vision systems, was $516 million (Brody, 1987).

TABLE 1-4 U. S. ROBOT MARKET (1986)

Application	Percent
Material handling	24.4
Spot welding	16.5
Arc welding	14.5
Spray painting and finishing	12.4
Mechanical assembly	6.2
Electronic assembly	4.8
Material removal	4.5
Inspection and testing	2.9
Water jet cutting	2.7
Other	11.1

Traditional applications of material handling, welding, and spray painting and finishing continue to dominate. The market share of *assembly* applications, both mechanical and electrical, has grown steadily over the past decade, and there is clearly potential for more applications in this area. However, the general assembly problem has turned out to be quite challenging. The assembly process can be modeled as a sequence of carefully planned *collisions* between the manipulator and the objects in its workspace. The delicate motion control that is required for assembly tasks dictates the use of feedback from external sensors. This allows the robot to *adapt* its motion in order to compensate for part tolerances and other uncertainties in the environment. Often customized fixtures and jigs are used to secure parts and present them to the robot at known positions and orientations (Boyes, 1985). This has the effect of reducing uncertainty, but it can be an expensive approach.

The world population of installed robots as of the end of 1984 (Wolovich, 1987) was reported to be approximately 68,000. The country that has the largest population of industrial robots is Japan; it is followed by the United States, West Germany, and Sweden, as can be seen in Table 1-5. If the robot population figures in Table 1-5 are normalized by the human population of the country, the leading country in per capita robot population is Sweden.

TABLE 1-5 DISTRIBUTION OF WORLD ROBOT POPULATION (1984)

Country	Percent
Japan	44.1
U.S.A.	22.1
West Germany	8.8
Sweden	7.4
France	2.9
Great Britain	2.9
Italy	2.2
Others	9.6

1-4 ROBOT SPECIFICATIONS

While the drive technologies, work-envelope geometries, and motion control methods provide convenient ways to broadly classify robots, there are a number of additional characteristics that allow the user to further specify robotic manipulators. Some of the more common characteristics are listed in Table 1-6.

TABLE 1-6 ROBOT CHARACTERISTICS

Characteristic	Units
Number of axes	—
Load carrying capacity	kg
Maximum speed, cycle time	mm/sec
Reach and stroke	mm
Tool orientation	deg
Repeatability	mm
Precision and accuracy	mm
Operating environment	—

1-4-1 Number of Axes

Each robotic manipulator has a number of *axes* about which its links rotate or along which its links translate. Usually, the first three axes, or major axes, are used to establish the position of the wrist, while the remaining axes are used to establish the orientation of the tool or gripper, as shown in Table 1-7. Since robotic manipulation is done in three-dimensional space, a *six-axis* robot is a *general* manipulator in the sense that it can move its tool or hand to both an arbitrary position and an arbitrary orientation within its workspace. The mechanism for opening and closing the fingers

TABLE 1-7 AXES OF A ROBOTIC MANIPULATOR

Axes	Type	Function
1–3	Major	Position the wrist
4–6	Minor	Orient the tool
7–*n*	Redundant	Avoid obstacles

or otherwise activating the tool is *not* regarded as an independent axis, because it does not contribute to either the position or the orientation of the tool. Practical industrial robots typically have from four to six axes. Of course, it is possible to have manipulators with more than six axes. The *redundant* axes can be useful for such things as reaching around obstacles in the workspace or avoiding undesirable geometrical configurations of the manipulator.

1-4-2 Capacity and Speed

Load-carrying capacity varies greatly between robots. For example, the Minimover 5 Microbot, an educational table-top robot, has a load-carrying capacity of 2.2 kg. At the other end of the spectrum, the GCA-XR6 Extended Reach industrial robot has a load-carrying capacity of 4928 kg (Roth, 1983–1984). The maximum *tool-tip speed* can also vary substantially between manipulators. The Westinghouse Series 4000 robot has a tool-tip speed of 92 mm/sec, while the Adept One SCARA robot has a tool-tip speed of 9000 mm/sec (Roth, 1983–1984). A more meaningful measure of robot speed may be the *cycle time*, the time required to perform a periodic motion similar to a simple pick-and-place operation. The Adept One SCARA robot carrying a 2.2-kg payload along a 700-mm path that consists of six straight-line segments has a cycle time of 0.9 sec. Thus the average speed over a cycle is 778 mm/sec, considerably less than the 9000 mm/sec maximum tool-tip speed.

Although the load-carrying capacities and maximum operating speeds of robots vary by several orders of magnitude, it is, of course, the *mix* of characteristics that is important when selecting a robot for a particular application. In some cases, a large load-carrying capacity may not be necessary, while in other cases accuracy may be more important than speed. Clearly, there is no point in paying for additional characteristics that are not relevant to the class of applications for which the robot is intended.

1-4-3 Reach and Stroke

The reach and the stroke of a robotic manipulator are rough measures of the size of the work envelope. The *horizontal reach* is defined as the maximum radial distance the wrist mounting flange can be positioned from the vertical axis about which the robot rotates. The *horizontal stroke* represents the total radial distance the wrist can travel. Thus the horizontal reach minus the horizontal stroke represents the minimum radial distance the wrist can be positioned from the base axis. Since this distance is nonnegative, we have:

$$\text{Stroke} \leq \text{reach} \tag{1-4-1}$$

For example, the horizontal reach of a cylindrical-coordinate robot is the radius of the outer cylinder of the work envelope, while the horizontal stroke is the difference between the radii of the concentric outer and the inner cylinders, as shown in Fig. 1-8.

The *vertical reach* of a robot is the maximum elevation above the work surface that the wrist mounting flange can reach. Similarly, the *vertical stroke* is the total vertical distance that the wrist can travel. Again, the vertical stroke is less than or

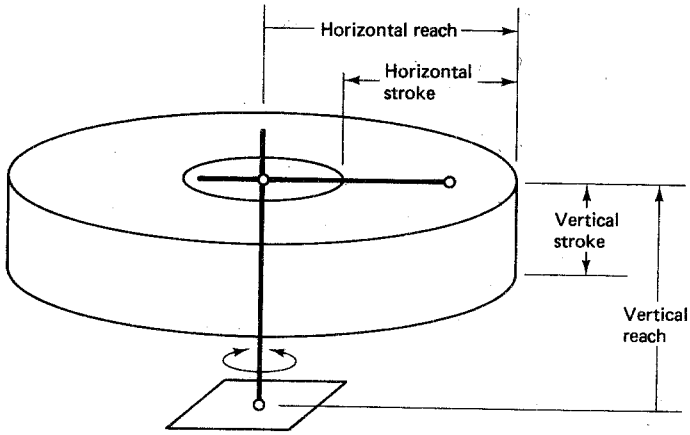


Figure 1-8 Reach and stroke of a cylindrical robot.

equal to the vertical reach. For example, the vertical reach of a cylindrical robot will be larger than the vertical stroke if the range of travel of the second axis does not allow the wrist to reach the work surface, as shown in Fig. 1-8. One of the useful characteristics of articulated robots lies in the fact that they often have *full* work envelopes, in the sense that the stroke equals the reach. However, this feature gives rise to a need for programming safeguards, because an articulated robot can be programmed to *collide* with itself or the work surface.

1-4-4 Tool Orientation

While the three major axes of a robot determine the shape of the work envelope, the remaining axes determine the kinds of orientation that the tool or hand can assume. If three independent minor axes are available, then *arbitrary* orientations in a three-dimensional workspace can be obtained. A number of conventions are used in the robotics literature to specify tool orientation (Fu et al., 1987). The tool orientation convention that will be used here is the *yaw-pitch-roll* (YPR) system. Yaw, pitch, and roll angles have long been used in the aeronautics industry to specify the orientation of aircraft. They can also be used to specify tool orientation, as shown in Fig. 1-9.

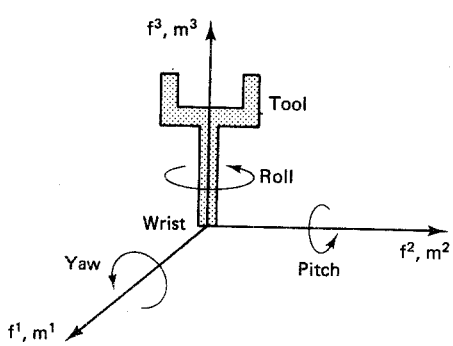


Figure 1-9 Yaw, pitch, and roll of tool.

To specify tool orientation, a mobile *tool coordinate frame* $M = \{m^1, m^2, m^3\}$ is attached to the tool and moves with the tool, as shown in Fig. 1-9. Here m^3 is aligned with the principal axis of the tool and points away from the wrist. Next, m^2 is parallel to the line followed by the fingertips of the tool as it opens and closes. Finally, m^1 completes the right-handed tool coordinate frame M.

By convention, the yaw, pitch, and roll motions are performed in a specific order about a set of fixed axes. Initially, the mobile tool frame M starts out coincident with a fixed *wrist coordinate frame* $F = \{f^1, f^2, f^3\}$ attached at the end of the forearm. First the yaw motion is performed by rotating the tool about wrist axis f^1. Next, the pitch motion is performed by rotating the tool about wrist axis f^2. Finally, the roll motion is performed by rotating the tool about wrist axis f^3. In each case positive angles correspond to *counterclockwise* rotations as seen looking along the axis back toward the origin.

The order in which the yaw, pitch, and roll motions are performed is significant because it affects the final orientation of the tool. For example, a yaw of $\pi/2$ followed by a pitch of $\pi/2$ yields a different final tool orientation from the orientation produced by a pitch of $\pi/2$ followed by a yaw of $\pi/2$. Thus, implicit in the YPR convention is the *order* of rotations summarized in Table 1-8.

TABLE 1-8 YAW, PITCH, AND ROLL MOTION

Operation	Description	Axis
1	Yaw	f^1
2	Pitch	f^2
3	Roll	f^3

An alternative way to specify the YPR motions is to instead perform the rotations in reverse order about the axes of the mobile tool frame M rather than the fixed wrist frame F. That is, first a roll motion is performed about m^3, then a pitch motion is performed about m^2, and finally a yaw motion is performed about m^1. This is equivalent to performing the rotations about the axes of the fixed wrist frame F in the original YPR order, in the sense that the tool ends up at the same orientation. For this reason, the YPR system is often referred to as the RPY system. A sequence of rotations about the mobile frame M axes is often easier to visualize, particularly when the angles are not multiples of $\pi/2$.

Definition 1-4-1: Spherical Wrist. A robot has a *spherical wrist* if and only if the axes used to orient the tool intersect at a point.

The robot wrist shown in Fig. 1-9 is an example of a spherical wrist because the three axes used to orient the tool, $\{m^1, m^2, m^3\}$, intersect at a point. More generally, if a robot possesses fewer than three axes to orient the tool, then it can still have a spherical wrist. However, in these cases the spherical wrist will not be a general spherical wrist, because, with fewer than three axes, some orientations of the tool cannot be achieved. Spherical wrists are useful because they simplify the mathematical descriptions of robotic manipulators. In particular, the larger n-axis problem can be partitioned into a two simpler problems, a 3-axis problem and an $(n - 3)$-

axis problem (Pieper, 1968). Many industrial robots are designed with spherical wrists.

1-4-5 Repeatability, Precision, and Accuracy

Repeatability is another important design characteristic. *Repeatability* is a measure of the ability of the robot to position the tool tip in the same place repeatedly. On account of such things as backlash in the gears and flexibility in the links, there will always be some repeatability error, perhaps on the order of a small fraction of a millimeter for a well-designed manipulator (Groover et al., 1986).

Closely related to the notion of repeatability is the concept of precision. The *precision* of a robotic manipulator is a measure of the spatial resolution with which the tool can be positioned within the work envelope. For example, if the tool tip is positioned at point A in Fig. 1-10 and the next closest position that it can be moved to is B, then the precision along that dimension is the distance between A and B.

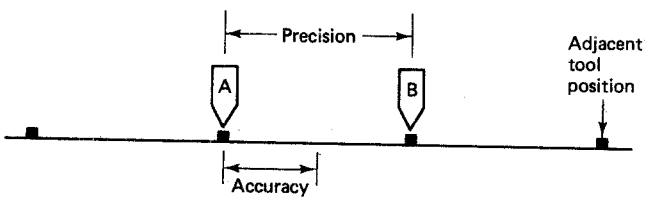


Figure 1-10 Adjacent tool positions.

More generally, in three-dimensional space the robot tool tip might be positioned anywhere on a three-dimensional *grid* of points within the workspace. Typically, this grid or lattice of points where the tool tip can be placed is not uniform. The overall precision of the robot is the maximum distance between neighboring points in the grid. For example, an interior grid point of a Cartesian-coordinate robot will have eight neighbors in the horizontal plane plus nine neighboring grid points in the planes above and below. One of the virtues of a Cartesian-coordinate robot is that the precision is *uniform* throughout the work envelope; the state of one axis does not influence the precision along another axis. However, this is not the case for other types of robots, as can be seen in Table 1-9, where the overall precision is decomposed into horizontal and vertical components.

For the case of a cylindrical-coordinate robot, the horizontal precision is highest along the inside surface of the work envelope and lowest along the outside surface. This is because the arc length swept out by an incremental movement $\Delta\phi$ in

TABLE 1-9 HORIZONTAL AND VERTICAL PRECISION

Robot Type	Horizontal Precision	Vertical Precision
Cartesian	Uniform	Uniform
Cylindrical	Decreases radially	Uniform
Spherical	Decreases radially	Decreases radially
SCARA	Varies	Uniform
Articulated	Varies	Varies

the base joint is proportional to the radial position r, as can be seen in Fig. 1-11. Note how the shape of a horizontal grid element is not square and grid elements near the outside surface of the work envelope are larger than grid elements near the inside surface of the work envelope.

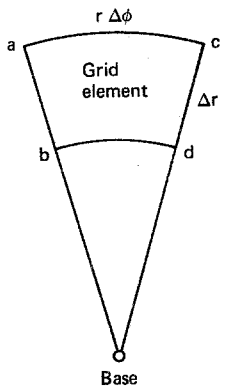


Figure 1-11 Horizontal precision of a cylindrical robot.

Here the radial precision is Δr, while the angular precision about the base expressed in terms of the arc length is $r\ \Delta\phi$. In the worst case, the angular precision is $R\ \Delta\phi$, where R is the horizontal reach of the robot. Thus the overall horizontal precision in this case is approximately

$$\Delta h \approx [(\Delta r)^2 + (R\ \Delta\phi)^2]^{1/2} \tag{1-4-2}$$

Here we have assumed that $\Delta\phi$ and Δr are sufficiently small and that the angle *bac* in Fig. 1-11 is approximately $\pi/2$. If Δz represents the vertical precision of a cylindrical-coordinate robot, then the total robot precision is

$$\Delta p \approx [(\Delta r)^2 + (R\ \Delta\phi)^2 + (\Delta z)^2]^{1/2} \tag{1-4-3}$$

For a spherical-coordinate robot, both the horizontal precision and the vertical precision are highest along the inside surface and lowest along the outside surface of the work envelope. For SCARA robots, the horizontal precision varies, but the vertical precision is uniform. It is this feature, together with its open work envelope, that makes the SCARA robot useful for light assembly work such as the vertical insertion of electrical components into circuit boards. With articulated robots, both the horizontal precision and the vertical precision vary over the work envelope.

An alternative way to specify the precision of a robot is to specify the precision of the individual joints. Clearly this is easier to analyze, although it is perhaps less useful. The precision or spatial resolution of an individual joint is affected by the drive system, the position sensors, and the power transmission mechanism, including gears, sprockets, and cables. For example, suppose the reference position signal is generated internally in a control computer and then sent to an external analog feedback position control system through an n-bit digital-to-analog converter (DAC). If the range of load shaft positions attainable goes from ϕ_{min} through ϕ_{max}, then the load shaft precision $\Delta\phi_{DAC}$ in this case is given by:

$$\Delta\phi_{DAC} = \frac{\phi_{max} - \phi_{min}}{2^n} \text{ radians per count} \tag{1-4-4}$$

On many robots, *incremental* encoders are used to determine shaft position rather than *absolute* encoders such as potentiometers. Incremental encoders typically consist of a slotted disk with infrared emitter-detector pairs. As the shaft turns, the infrared beams are either interrupted or not, depending upon the position of the slotted disk. If two or more emitter-detectors are used, then both the *direction* and the *magnitude* of the rotary motion can be determined. This can be seen in Fig. 1-12, where a slotted disk with a pair of slots arranged in phase quadrature is displayed. Here a broken beam is represented by a 1, while an unbroken beam is represented by a 0. Note that there are four possible states. Since only one bit changes between adjacent states, the overlapping slots generate what is called a *gray code*. If the encoder output is sampled sufficiently fast, then the direction of the movement can be inferred by comparing the old state with the new state. The counter which stores the accumulated shaft position is then either incremented or decremented as appropriate.

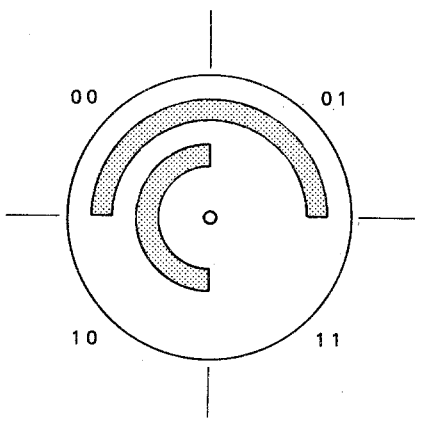


Figure 1-12 Slotted disk with 2-bit gray code.

More generally, if d emitter-detector pairs are used, and if there are k slots around the circumference of the disk, then the precision with which the shaft position can be measured is $2\pi/(k2^d)$ radians per count. The encoded disk is usually mounted on the high-speed shaft of the motor, rather than the load shaft. If the gear reduction ratio from the high-speed shaft to the load shaft is $m:1$, then the overall incremental load shaft precision $\Delta\phi_{\text{inc}}$ is

$$\Delta\phi_{\text{inc}} = \frac{\pi}{km\,2^{d-1}} \text{ radians per count} \tag{1-4-5}$$

The *accuracy* of a robotic manipulator is a measure of the ability of the robot to place the tool tip at an arbitrarily prescribed location in the work envelope. Robotic arm accuracy is not nearly as easy to analyze as precision. Accuracy depends, for example, on such things as gear backlash and elastic deformations in the links due to loading. However, a simple bound can be placed on the error or inaccuracy, in terms of the precision, namely:

$$\text{Error} \geq \frac{\text{precision}}{2} \tag{1-4-6}$$

Clearly, if the precision of the robot dictates that only a discrete grid of points can be visited within the work envelope, then the accuracy with which an arbitrary point can be visited is, at best, half of the grid size, as shown in Fig. 1-10. In general, the accuracy will not be that good. For example, the robot may gradually drift out of calibration with prolonged operation. To maintain reasonable accuracy, robots are periodically reset or zeroed to a standard *hard home* position using limit switch sensors.

Although robotic manipulators are not nearly as accurate as their more expensive and more specialized hard automation counterparts, they do provide increased flexibility. It is one of the great challenges of robotics to make use of this increased flexibility to compensate for less than perfect accuracy. In particular, clever algorithms and control strategies are needed that make use of external sensors to compensate for uncertainties in the environment and the uncertainties in the exact position of the robot itself.

1-4-6 Operating Environment

The operating environments within which robots function depend on the nature of the tasks performed. Often robots are used to perform jobs in harsh, dangerous, or unhealthy environments. Examples include the transport of radioactive materials, spray painting, welding, and the loading and unloading of furnaces. For these applications the robot must be specifically designed to operate in extreme temperatures and contaminated air. In the case of paint spraying, a robotic arm might be *clothed* in a shroud in order to minimize the contamination of its joints by the airborne paint particles.

At the other extreme of operating environments are clean room robots. Clean room robots are used in the semiconductor manufacturing industry for such tasks as the handling of silicon wafers and photomasks. Specialized equipment has to be loaded and unloaded at the various clean room processing stations. Here the robot is operating in an environment in which parameters such as temperature, humidity, and airflow are carefully controlled. In this case it is the robot itself that must be designed to be very clean so as to minimize the contamination of work pieces by submicron particles. Some clean room robots are evacuated internally with suction in order to *scavange* particles generated by friction surfaces. Others use special nonshedding materials and employ magnetic washers to hold ferromagnetic lubricant in place.

1-4-7 An Example: Rhino XR-3

In recent years a number of educational table-top robots have appeared in the marketplace. Many educational robots employ open-loop position control using DC stepper motors. Some table-top educational robots are also available which use closed-loop position control with DC servomotors and incremental encoders. An example of the latter type of educational robot is the Rhino XR-3 robot, pictured in Fig. 1-13.

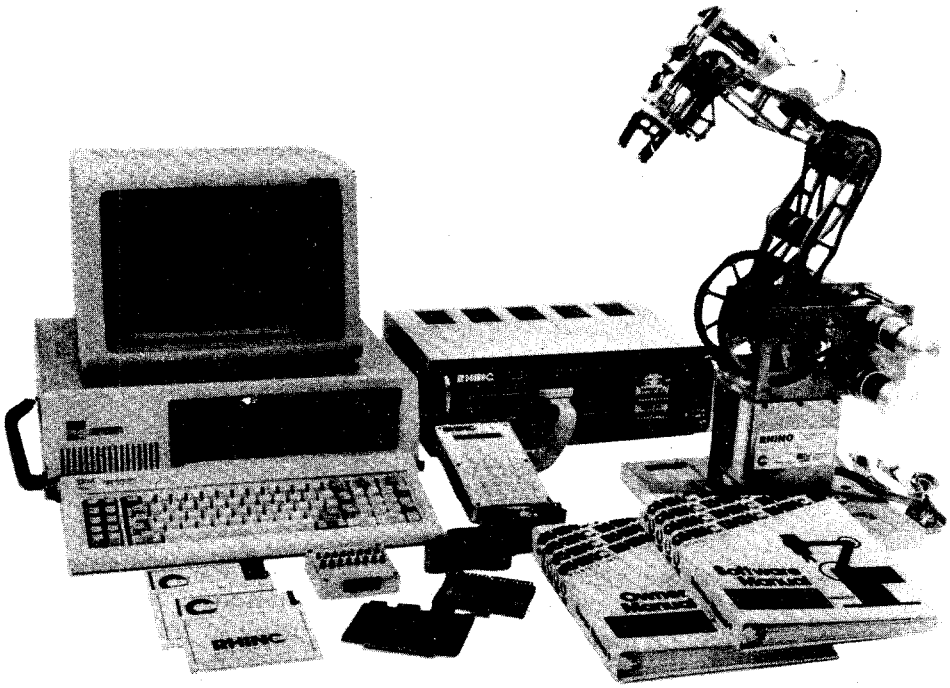


Figure 1-13 An educational robot. (Courtesy of Rhino Robots, Inc., Champaign, IL. Robotic arm developed and manufactured in the U.S.A.)

In terms of broad classification, the Rhino XR-3 robotic manipulator is a five-axis electric-drive articulated-coordinate robot with point-to-point motion control (Hendrickson and Sandhu, 1986). The specifications of the Rhino XR-3 are summarized in Table 1-10. The load-carrying capacity, maximum tool-tip speed, and repeatability are rough estimates based on use of this robot in an instructional laboratory. The reach and stroke specifications are measured with respect to the tool tip assuming that standard fingers are used for the tool.

Another useful specification for the Rhino robot is the load shaft precision for each of the five axes. Each axis is driven by a DC servomotor that has an in-

TABLE 1-10 RHINO XR-3 SPECIFICATIONS

Characteristic	Value	Units
Number of axes	5	—
Load-carrying capacity	0.5	kg
Maximum tool-tip speed	25.0	cm/sec
Horizontal reach and stroke	62.23	cm
Vertical reach and stroke	88.27	cm
Tool pitch range	270.0	deg
Tool roll range	infinite	deg
Repeatability	±0.1	cm

cremental encoder attached to the high-speed shaft. The encoders resolve the high-speed position to 60 degrees. Since each motor has a built-in gear head with a turns ratio of 66.1 : 1 or 96 : 1, this results in a precision for each load shaft of 0.908 or 0.624 degrees per count. Finally, each joint has its own set of sprockets and chains which then determine the joint angle precision. The resulting precision in degrees per count for each joint is summarized in Table 1-11.

TABLE 1-11 LOAD SHAFT PRECISION OF THE RHINO XR-3 ROBOT

Motor	Joint	Precision
F	Base	0.2269
E	Shoulder	0.1135
D	Elbow	0.1135
C	Tool pitch	0.1135
B	Tool roll	0.3125

Units of precision: degrees per count.

The Rhino XR-3 robot is one of several robots that are used as vehicles for illustrating theoretical concepts covered in the remainder of this text. Direct and inverse kinematics, work-envelope calculations, trajectory planning, and control are examined. A series of laboratory projects designed around a robotic work cell which features a Rhino XR-3 robot and a Micron Eye camera are included in a separate Laboratory Manual that accompanies this text (Schilling and White, 1990). These laboratory programming projects are designed to play an integral part in any course using this text and are strongly recommended. If a Rhino XR-3 or other educational robot is not available, a three-dimensional *graphics robot simulator* program called SIMULATR that is supplied with the Laboratory Manual can be used as a substitute. The use of the graphics simulator for development of robot control programs is highly recommended even when a physical robot is available. This way robot control programs can be conveniently developed and debugged off-line with the final version of the program tested on an actual robot.

1-5 NOTATION

Robotics is a broad interdisciplinary field that encompasses a variety of traditional topics from electrical engineering, mechanical engineering, and computer science. This breadth makes the field quite exciting, but at the same time it poses a challenge when it comes to developing a clear and consistent notation. Since notation and terminology play an important role in any detailed treatment of robotics, we pause at this point to discuss and summarize the notational conventions that will be used in the remainder of this text.

Scalars, vectors, matrices, and sets are used to simplify the presentation and make it more concise. To distinguish vectors and matrices from scalars, boldface print is traditionally employed. Although this approach is effective for printed material, the use of boldface characters is, at best, inconvenient when instructors are

writing on a blackboard or when students are taking notes in a class notebook. One common alternative for these media is the underlining convention whereby vectors are underlined once and matrices twice. However, repeated underlining becomes very cumbersome when vectors and matrices are used extensively.

The general convention employed here is to instead regard *all* mathematical quantities as either column vectors, matrices, or sets. *Column vectors* are denoted with *lowercase* letters, both English and Greek, while *matrices* and *sets* are denoted with *uppercase* letters. Single *subscripts* denote scalar components of column vectors, whereas double subscripts denote scalar components of matrices. In the case of matrices, the first subscript is the index of the row, while the second is the index of the column. These general notational conventions are summarized in Table 1-12.

TABLE 1-12 NOTATIONAL CONVENTIONS

Entity	Notation	Examples
Scalars	Subscripted	$a_1, a_2, \alpha_1, \alpha_2, A_{11}, A_{12}$
Column vectors	Lowercase	$a, b, c, \alpha, \beta, \gamma, x^1, x^2$
Matrices, sets	Uppercase	$A, B, C, D, \Gamma, \Omega, \Psi, \Upsilon$

Note that scalars can be regarded as important special cases of column vectors, that is, column vectors with only one component. Similarly, vectors are themselves special cases of matrices, that is, matrices with only one row or only one column.

Vectors. A column vector is represented by arranging its components in an $n \times 1$ array and enclosing them in *square brackets* as follows:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{1-5-1}$$

The superscript T is reserved to denote the *transpose* of a vector or, more generally, of a matrix. The transpose is obtained by interchanging the rows with the columns. Thus the transpose of an $n \times 1$ column vector is a $1 \times n$ row vector, and conversely. The following is therefore a space-saving way of writing a column vector as a transposed row vector:

$$x = [x_1, x_2, x_3]^T \tag{1-5-2}$$

Individual *superscripts* are used as *indices* to distinguish members of a set of vectors. In those cases where a superscript might be confused with taking a quantity to a power, parentheses are used to remove the ambiguity. For example $(x_1)^2$ is used to denote the square of the first component of the vector x, while x_1^2 denotes the first component of the second member of a set of vectors $\{x^1, x^2, \ldots\}$.

Sets. While square brackets are reserved for enclosing components of vectors and matrices, curly brackets, or *braces*, are used to enclose members of a set. Thus a set Γ of vectors $x^1, x^2, \ldots, x^n$ is represented as follows:

$$\Gamma = \{x^1, x^2, \ldots, x^n\} \tag{1-5-3}$$

We often need to specify a set of vectors satisfying a certain *property P*—for example, the set of all solutions to a particular equation or perhaps the set of all vectors whose first component is zero. In these cases the set of all vectors belonging to subset U that also satisfy property P is represented as follows:

$$\Gamma = \{x \in U: P(x)\}. \tag{1-5-4}$$

Here $\in$ is the set membership symbol, and $P(x)$ is a property of x. For example, $P(x)$ might represent one or more equality or inequality constraints involving x or its components.

Matrices. Just as a vector can be represented as a one-dimensional array of scalar components, as in Eq. (1-5-1), a matrix can be represented by a two-dimensional array of scalar components as follows:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \tag{1-5-5}$$

It is also useful to represent a matrix in terms of its columns. The convention used here is that the corresponding lowercase letter with a superscript index is used to denote a column of a matrix. Thus a^j denotes the *jth* column of the matrix A, in which case $A_{kj} = a_k^j$, for $1 \leq k \leq m$ and $1 \leq j \leq n$. The entire $m \times n$ matrix A can be written in terms of its n columns as follows:

$$A = [a^1, a^2, \ldots, a^n] \tag{1-5-6}$$

The matrix A can be viewed as a linear transformation or mapping from the *vector space* $\mathbf{R}^n$ of column vectors with n real components to the vector space $\mathbf{R}^m$ of column vectors with m real components. Every $m \times n$ matrix A has a *null space* $N(A)$ and a *range space R(A)* associated with it; these are defined as follows:

$$N(A) = \{x \in \mathbf{R}^n: Ax = 0\} \tag{1-5-7}$$

$$R(A) = \{Ax \in \mathbf{R}^m: x \in \mathbf{R}^n\} \tag{1-5-8}$$

The null space of A is the subspace of $\mathbf{R}^n$ consisting of the set of all vectors $x \in \mathbf{R}^n$ which are mapped into the zero vector in $\mathbf{R}^m$ by the transformation A. Thus the null space is the *inverse image* of the zero vector in $\mathbf{R}^m$ under the transformation A. Similarly, the range space of A is the subspace of $\mathbf{R}^m$ consisting of all vectors y such that $y = Ax$ for at least one $x \in \mathbf{R}^n$. Thus the range space $R(A)$ is the *image* of $\mathbf{R}^n$ under the transformation A. A pictorial representation of the null and range spaces of a matrix A is shown in Fig. 1-14.

The dimension of $N(A)$ is called the *nullity* of A, and the dimension of $R(A)$ is called the *rank* of A. One of the fundamental results of linear algebra is that the dimension of the null space plus the dimension of the range space equals the dimension of the space over which the transformation A is defined. Therefore:

$$\text{nullity } (A) + \text{rank } (A) = n \tag{1-5-9}$$

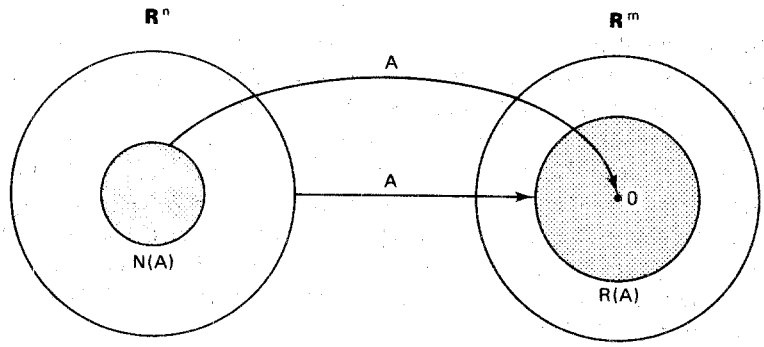


Figure 1-14 The range and null space of an $m \times n$ matrix A.

The rank of A is equal to the number of linearly independent rows of A, and it is also equal to the number of linearly independent columns of A. If the rank of the matrix A is as large as possible, then A is said to be of *full rank*. A square matrix ($m = n$) is *invertible* if and only if it is of full rank. We denote the *inverse* of A as A^{-1}.

Certain special matrices have symbols reserved for them. These include the *zero matrix*, which is denoted by 0, and the *identity matrix*, which is denoted by I. The zero matrix is simply a matrix of 0s, while the identity matrix is a square matrix that has 0s everywhere except along the principal diagonal, where it has 1s.

Coordinate transformations. To model the motion of a robotic arm, coordinate frames are attached to various parts of the arm and to sensors and objects in the workspace. In particular, the following right-handed orthonormal coordinate frames are attached to the links of the robot, starting with link 0, which is the base, and ending with link n, which is the tool:

$$L_k = \{x^k, y^k, z^k\}, \qquad 0 \leq k \leq n \tag{1-5-10}$$

It is often necessary to transform or map the coordinates of a point with respect to one frame into its coordinates with respect to another frame. The matrix T is reserved to denote a general *coordinate transformation matrix*. Typically, T will have a superscript index or identifier which specifies the source coordinate frame and a subscript index or identifier which specifies the destination or reference coordinate frame. For example, $T_{\text{base}}^{\text{tool}}$ denotes the coordinate transformation matrix which transforms robot tool coordinates into robot base coordinates. More generally, T_j^k transforms frame L_k coordinates into frame L_j coordinates.

Some authors of robotics texts use either presuperscripts or presubscripts, or both, such as ${}^kT_j$ and ${}_{\text{base}}^{\text{tool}}T$, for coordinate transformation matrices (Paul, 1981; Craig, 1986). We will refrain from using any presuperscripts or presubscripts, in an effort to simplify the notation somewhat. The price we pay is that in some cases there may be ambiguity as to whether or not a superscript should be interpreted as taking a matrix to a power. In these cases we again use parentheses to remove the ambiguity. Thus $(T_1)^2$ is the square of the matrix T_1, while T_1^2 is the matrix which transforms frame L_2 coordinates into frame L_1 coordinates.

Special symbols. A number of special notational symbols are used throughout the text. In each case they are defined in the text where they *first appear* unless they are common, widely accepted symbols. In any event, a list of symbols can be found in Appendix 3. This list contains brief descriptions of special symbols used repeatedly throughout the text. Symbols which are used only infrequently, such as for a temporary variable in a derivation or proof, are not included in this list.

Simple trigonometric expressions appear repeatedly in the analysis of robotic arms. Expressions involving sums and products of sines and cosines of joint angles can be shortened and simplified if a standard trigonometric shorthand is employed as summarized in Table 1-13.

TABLE 1-13 TRIGONOMETRIC SHORTHAND

Symbol	Meaning
$C\phi$	$\cos \phi$
$S\phi$	$\sin \phi$
$V\phi$	$1 - \cos \phi$
C_k	$\cos \theta_k$
S_k	$\sin \theta_k$
C_{kj}	$\cos (\theta_k + \theta_j)$
S_{kj}	$\sin (\theta_k + \theta_j)$
C_{k-j}	$\cos (\theta_k - \theta_j)$
S_{k-j}	$\sin (\theta_k - \theta_j)$

Here θ_k denotes the joint angle for the kth joint of a robotic arm. Clearly the notation can be extended in the obvious way to include sums or differences of three or more joint angles. The joint angle θ_k is often replaced by the variable q_k, which denotes the *joint variable* for the kth joint. That is, q_k and θ_k can be used interchangeably in Table 1-13.

1-6 PROBLEMS

1-1. What is the essential feature that distinguishes soft automation from hard automation?

1-2. Which types of drive technologies would be most suitable for the following applications:
(a) Unload 1000-kg items from a die casting machine
(b) Install integrated circuit chips on a printed circuit board

1-3. What is the minimum number of axes required for a robot to insert and tighten four nuts (from above) on four vertical bolts as shown in Fig. 1-15? The nuts are available from a vertical spring-activated parts feeder. Explain why each axis is needed.

1-4. Suppose a cylindrical-coordinate robot has a vertical reach of 480 mm and a vertical stroke of 300 mm. How far off the floor do parts have to be raised in order to be reachable by the robot?

1-5. The base joint of a cylindrical-coordinate robot is driven by a 12-bit digital-to-analog

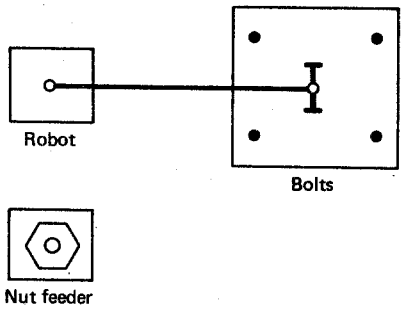


Figure 1-15 Nut tightening task.

converter (DAC) and has a swing range of 360 degrees. The radial axis is driven by an 8-bit DAC and has a horizontal reach of 300 mm and a horizontal stroke of 200 mm. Finally, the vertical axis is driven by a 10-bit DAC and has a vertical reach of 480 mm and a vertical stroke of 360 mm.

(a) What is the volume of the work envelope?
(b) What is the vertical precision?
(c) What is the radial precision?
(d) What is the angular precision about the base?
(e) What is the horizontal precision?
(f) What is the total precision?

1-6. For a spherical-coordinate robot, where within the work envelope is the tool-tip precision the highest?

1-7. For what type of robot is the precision uniform throughout the work envelope? For which robots is the vertical precision uniform?

1-8. An incremental shaft encoder with 2 emitter-detector pairs and 12 slots around the circumference is used to monitor the angular position of a high-speed motor shaft. The precision of the load shaft is measured and found to be 0.05 degrees per count. What is the gear ratio between the high-speed shaft and the load shaft?

1-9. Consider the robotic tool shown in Fig. 1-16. Sketch the tool position after each intermediate position of the following YPR operation: yaw $\pi/2$, pitch $-\pi/2$, roll $\pi/2$.

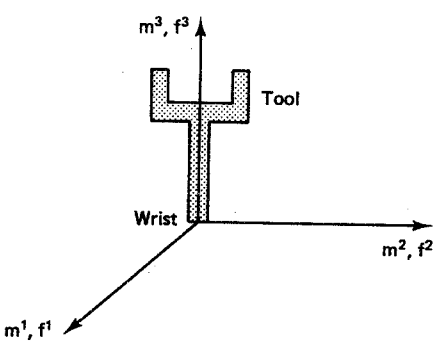


Figure 1-16 Yaw, pitch, and roll motion.

1-10. Verify that if the YPR operations in Prob. 1-9 are performed in reverse order and if the rotations are taken about the axes of the mobile frame M rather than the fixed frame F, then the final tool position is the same. Are the intermediate tool positions the same?

REFERENCES

ASIMOV, I. (1950). *I, Robot*, Fawcett Publications: Greenwich, Conn.

BOYES, W. E. (1985). *Low-Cost Jigs, Fixtures, and Gages for Limited Production*, SME Publications: Dearborn, Mich.

BRODY, H. (1987). "U.S. robot makers try to bounce back," *High Technology Business*, October, pp. 18–24.

CRAIG, J. J. (1986). *Introduction to Robotics: Mechanics & Control*, Addison-Wesley: Reading, Mass.

DORF, R. C. (1983). *Robotics and Automated Manufacturing*, Reston: Reston, Va.

FU, K. S., R. C. GONZALEZ, AND C. S. G. LEE (1987). *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill: New York.

GROOVER, M. P., M. WEISS, R. N. NAGEL, AND N. G. ODREY (1986). *Industrial Robotics*, McGraw-Hill: New York.

HENDRICKSON, T., AND H. SANDHU (1986). *XR-3 Robot Arm, Mark III 8 Axis Controller: Owner's Manual*, Version 3.00, Rhino Robots, Inc.: Champaign, Ill.

PAUL, R. (1981). *Robot Manipulators: Mathematics, Programming, and Control*, MIT Press: Cambridge, Mass.

PIEPER, D. L. (1968). "The kinematics of manipulators under computer control," *AIM 72*, Stanford AI Laboratory, Stanford Univ.

ROBOT INSTITUTE OF AMERICA (1982). *Worldwide Robotics Survey and Directory*, Dearborn, Mich.

ROTH, B. (1983–1984). "A table of contemporary manipulator devices," *Robotics Age*, July 1983, pp. 38–39; September 1983, pp. 30–31, January 1984, pp. 34–35.

SCHILLING, R. J., AND R. B. WHITE (1990). *Robotic Manipulation: Programming and Simulation Studies*, Prentice Hall, Englewood Cliffs, N.J.

WOLOVICH, W. A. (1987). *Robotics: Basic Analysis and Design*, Holt, Rinehart & Winston: New York.

2

Direct Kinematics: The Arm Equation

A robotic manipulator can be modeled as a chain of rigid bodies called *links*. The links are interconnected to one another by *joints*, as shown in Fig. 2-1. One end of the chain of links is fixed to a base, while the other end is free to move. The mobile end has a flange, or face plate, with a *tool*, or *end-effector*, attached to it. There are typically two types of joints which interconnect the links: revolute joints, which exhibit rotational motion about an axis, and prismatic joints, which exhibit sliding motion along an axis.

The objective is to control both the position and the orientation of the tool in three-dimensional space. The tool, or end-effector, can then be programmed to follow a planned trajectory so as to manipulate objects in the workspace. In order to program the tool motion, we must first formulate the relationship between the joint variables and the position and the orientation of the tool. This is called the *direct kinematics problem*, which we define formally as follows:

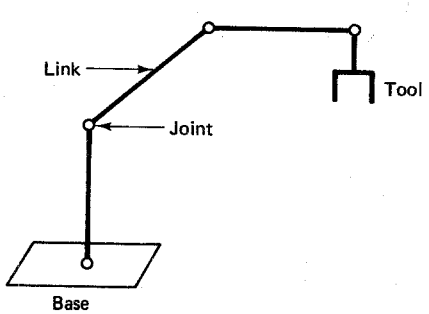


Figure 2-1 Robotic manipulator modeled as a chain of links.

Problem: Direct Kinematics. Given the vector of joint variables of a robotic manipulator, determine the position and orientation of the tool with respect to a coordinate frame attached to the robot base.

To develop a concise formulation of a general solution to the direct kinematics problem, it is helpful to first briefly review some fundamental properties of vector spaces (Noble, 1969; Schilling and Lee, 1988). This is followed by a discussion of rotations, translations, and screw transformations between coordinate frames. Four-dimensional homogeneous coordinates are employed in order to develop a matrix representation of these operations. A systematic procedure for assigning coordinate frames to the links of a robotic manipulator is then presented (Denavit and Hartenberg, 1955), and this leads directly to the arm equation. The solution to the direct kinematics problem is illustrated using three robotic manipulators: the five-axis Rhino XR-3, the four-axis Adept One, and the six-axis Intelledex 660. These robots are used repeatedly as case study type examples throughout the remainder of the text.

2-1 DOT AND CROSS PRODUCTS

Vectors in *n-dimensional space* $\mathbf{R}^n$ can be thought of as arrows emanating from the origin, as shown in Fig. 2-2 for the case $n = 3$. The coordinates of the vector are then synonymous with the location of the tip of the arrowhead.

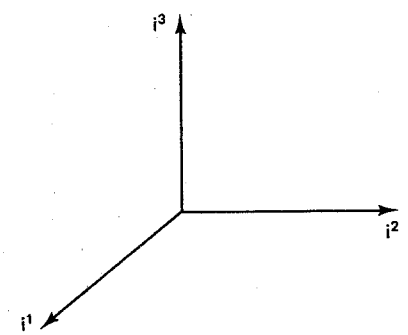


Figure 2-2 An orthonormal coordinate frame in $\mathbf{R}^3$.

The vectors pictured in Fig. 2-2 are special in the sense that *any* vector in the space $\mathbf{R}^3$ can be easily represented in terms of these vectors. To see this, it is useful to first introduce the notion of an inner product, or dot product, of two vectors.

Definition 2-1-1: Dot Product. The *dot product* of two vectors x and y in $\mathbf{R}^n$ is denoted $x \cdot y$ and is defined:

$$x \cdot y \triangleq \sum_{k=1}^{n} x_k y_k$$

The symbol $\triangleq$ should be read "equals by definition." The dot product is also called the *Euclidean inner product* in $\mathbf{R}^n$. The matrix transpose operation can be used to express the dot product more compactly as

$$x \cdot y = x^T y. \tag{2-1-1}$$

Here x^T denotes the row vector obtained by taking the transpose of the column vector x. The dot product in $\mathbf{R}^n$ has all the properties that are characteristic of inner products in general. In particular, the following properties are fundamental:

Proposition 2-1-1: Dot Product. Let $\{x, y, z\}$ be vectors $\mathbf{R}^n$ and let $\{\alpha, \beta\}$ be real scalars. Then the dot product has the following properties:

1. $x \cdot x \geq 0$
2. $x \cdot x = 0 \Leftrightarrow x = 0$
3. $x \cdot y = y \cdot x$
4. $(\alpha x + \beta y) \cdot z = \alpha(x \cdot z) + \beta(y \cdot z)$

These properties follow directly from Def. 2-1-1. Note that the first two properties indicate that the dot product of a vector with itself is always nonnegative and is zero only for the zero vector. The third property specifies that the dot product is a *commutative* function of its two arguments, that is, interchanging the order of x and y does not alter the result. Finally, the last and perhaps most important property is that the dot product is a *linear* function of its first argument x. Thus the dot product of the sum is the sum of the dot products. In view of the commutative property, the dot product is also a linear function of its second argument y.

The dot product of two vectors is a measure of the *orientation* between the two vectors. To see this, it is useful to first examine the concept of orthogonality.

Definition 2-1-2: Orthogonality. Two vectors x and y in $\mathbf{R}^n$ are *orthogonal* if and only if $x \cdot y = 0$.

For the Euclidean inner product or dot product in Def. 2-1-1, orthogonal vectors can be interpreted geometrically in three-dimensional space as *perpendicular* vectors. Thus the three vectors shown in Fig. 2-2, which correspond to the three columns of the identity matrix I, are mutually orthogonal. We therefore refer to them as an *orthogonal set*. Not only is the set of vectors in Fig. 2-2 an orthogonal set, it is also a complete set. A complete orthogonal set is a set of vectors with the property that the only vector orthogonal to every member of the set is the zero vector.

Definition 2-1-3: Completeness. An orthogonal set of vectors $\{x^1, x^2, \ldots, x^n\}$ in $\mathbf{R}^n$ is *complete* if and only if

$$y \cdot x^k = 0 \quad \text{for} \quad 1 \leq k \leq n \quad \Rightarrow \quad y = 0$$

The number of vectors it takes to form a complete orthogonal set for a vector space is called the *dimension* of the space. Thus $\mathbf{R}^3$ is a three-dimensional space and, more generally, $\mathbf{R}^n$ is an n-dimensional space. The vectors pictured in Fig. 2-2 are also special in one final sense. The *length* of each vector is unity, and so we refer to these vectors as *unit vectors*. The length or norm of an arbitrary vector in $\mathbf{R}^n$ can be defined in terms of the dot product as follows:

Definition 2-1-4: Norm. The *norm* of a vector x in $\mathbf{R}^n$ is denoted as $\|x\|$ and is defined:

$$\|x\| \triangleq (x \cdot x)^{1/2} = \left(\sum_{k=1}^{n} x_k^2 \right)^{1/2}$$

Thus the norm is an n-dimensional generalization of the absolute value function. A set of orthogonal unit vectors is referred to as an *orthonormal set*. The vectors shown in Fig. 2-2 are an example of a complete orthonormal set, or an *orthonormal coordinate frame*. It is a *right-handed* coordinate frame, because when the fingers of the right hand are curled from the first vector into the second vector, the thumb points in the direction of the third vector. One important result that will be useful when we examine rotating coordinate frames is the following geometric relationship between the dot product and the norm:

Proposition 2-1-2: Dot Product. Let x and y be arbitrary vectors in $\mathbf{R}^3$ and let θ be the angle from x to y. Then:

$$x \cdot y = \|x\| \, \|y\| \cos \theta$$

Hence the dot product can be interpreted as a measure of the *orientation* between two vectors. It is proportional to the cosine of the angle between the vectors, the constant of proportionality being the product of the lengths of the vectors. It follows from Prop. 2-1-2 and Def. 2-1-2 that the angle between orthogonal vectors is $\pi/2$ radians. Hence orthogonal vectors in $\mathbf{R}^3$ are perpendicular to one another.

There is another operation with vectors which is useful in the analysis of robotic manipulators, the cross product. Whereas the dot product of two vectors generates a scalar, the cross product of two vectors generates another vector that can be defined as follows:

Definition 2-1-5: Cross Product. The *cross product* of two vectors u and v in $\mathbf{R}^3$ is a vector $w = u \times v$ which is orthogonal to u and v in the right-handed sense and defined as follows:

$$w \triangleq \det \left\{ \begin{bmatrix} i^1 & i^2 & i^3 \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{bmatrix} \right\} = \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix}$$

Here the determinant notation is used in a formal sense to indicate how to evaluate the cross product. In this case the unit vectors $\{i^1, i^2, i^3\}$ are treated as if they were scalars for the purpose of computing the determinant. For the right-handed orthonormal coordinate frame shown in Fig. 2-2, $i^1 = i^2 \times i^3$, $i^2 = i^3 \times i^1$, and $i^3 = i^1 \times i^2$. Recall that the value of the dot product $u \cdot v$ depends upon the angle between the two vectors u and v. Similarly, the length of the cross-product vector $u \times v$ depends on the angle between the two vectors u and v, as can be seen from the following result:

Proposition 2-1-3: Cross Product. Let u and v be arbitrary vectors in $\mathbf{R}^3$ and let θ be the angle from u to v. Then:

$$\|u \times v\| = \|u\| \, \|v\| \sin \theta$$

Thus, whereas the dot product is proportional to the cosine of the angle between the vectors, the size of the cross product vector is proportional to the sine of the angle between the vectors. Refer to Fig. 2-3 for an illustration of two vectors u and v in $\mathbf{R}^3$ and their cross product.

Dot products and cross products are useful in developing concise formulations of the kinematics, statics, and dynamics of robotic manipulators. We begin with the kinematic analysis.

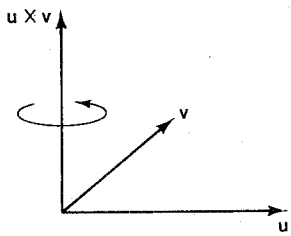


Figure 2-3 Cross product of u with v.

2-2 COORDINATE FRAMES

The inner product, or dot product, is useful because one can employ it to represent arbitrary vectors in terms of their projections onto subspaces generated by the members of a complete orthonormal set of vectors. In particular, suppose p is an arbitrary vector in the space $\mathbf{R}^n$, and suppose $X = \{x^1, x^2, \ldots, x^n\}$ is a complete orthonormal set for $\mathbf{R}^n$. We can then represent the vector p as a linear combination (weighted sum) of the vectors in X where the weighting coefficients are called the coordinates of p with respect to X.

Definition 2-2-1: Coordinates. Let p be a vector in $\mathbf{R}^n$ and let $X = \{x^1, x^2, \ldots, x^n\}$ be a complete orthonormal set for $\mathbf{R}^n$. Then the *coordinates* of p with respect to X are denoted $[p]^X$ and are defined implicitly by the equation:

$$p = \sum_{k=1}^{n} [p]_k^X x^k$$

The complete orthonormal set X is sometimes called an orthonormal *coordinate frame* or an orthonormal *basis* for the space $\mathbf{R}^n$. When the coordinate-frame vectors are orthonormal, the coordinates of a vector with respect to that coordinate frame are particularly easy to compute, as can be seen from the following fundamental result:

Proposition 2-2-1: Orthonormal Coordinates. Let p be a vector in $\mathbf{R}^n$ and let $[p]^X$ be the coordinates of p with respect to an orthonormal coordinate frame $X = \{x^1, x^2, \ldots, x^n\}$. Then the kth coordinate of p with respect to X is:

$$[p]_k^X = p \cdot x^k \qquad 1 \leq k \leq n$$

Proof. From Def. 2-2-1, it is sufficient to show that c is the zero vector where c is the difference or error:

$$c \triangleq p - \sum_{k=1}^{n} (p \cdot x^k) x^k$$

Using Def. 2-1-2, Def. 2-1-4, and the properties of the dot product summarized in Prop. 2-1-1, we have for each $1 \leq j \leq n$:

$$\begin{aligned} c \cdot x^j &= p \cdot x^j - \left[\sum_{k=1}^{n} (p \cdot x^k) x^k \right] \cdot x^j \\ &= p \cdot x^j - \sum_{k=1}^{n} (p \cdot x^k)(x^k \cdot x^j) \\ &= p \cdot x^j - \sum_{k=1}^{n} (p \cdot x^k) I_{kj} \\ &= p \cdot x^j - p \cdot x^j \\ &= 0 \end{aligned}$$

Thus c is orthogonal to every member of X. Because X is complete, it follows from Def. 2-1-3 that $c = 0$ and thus that Prop. 2-2-1 is valid.

We see that the kth coordinate of p with respect to an orthonormal coordinate frame X is simply the dot product of p with the kth member of the set X. The geometric interpretation of the relationship between a vector and its coordinates is shown in Fig. 2-4, where, for simplicity, the two-dimensional space $\mathbf{R}^2$ is used. Note how p is decomposed into the sum of its orthogonal projections onto the one-dimensional subspaces (lines) generated by x^1 and x^2. In view of the simple relationship between a vector and its orthonormal coordinates, *all* coordinate frames used in the remainder of the text are assumed to be right-handed orthonormal frames unless otherwise noted.

The solution to the direct kinematics problem in robotics requires that we represent the position and orientation of the mobile tool with respect to a coordinate frame attached to the fixed base. This involves a sequence of coordinate transformations from tool to wrist, wrist to elbow, elbow to shoulder, and so on. Each of these coordinate transformations can be represented by a matrix. To see this, it is useful to first examine a simple special case, the single-axis "robot" displayed in Fig. 2-5. Here a single revolute joint connects a fixed link (the base) to a mobile link (the body). The problem is to represent the position of a point p on the body with respect to a coordinate frame attached to the base. The coordinates of p in this case will vary as the joint angle θ changes. Let $F = \{f^1, f^2, f^3\}$ and $M = \{m^1, m^2, m^3\}$ be orthonormal coordinate frames attached to the fixed and the mobile links, respec-

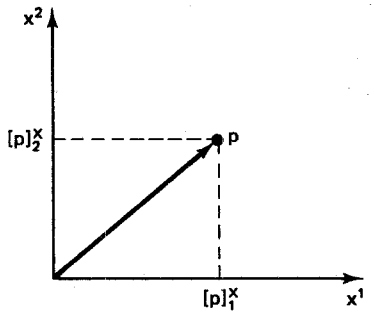


Figure 2-4 Coordinates of p with respect to the coordinate frame X.

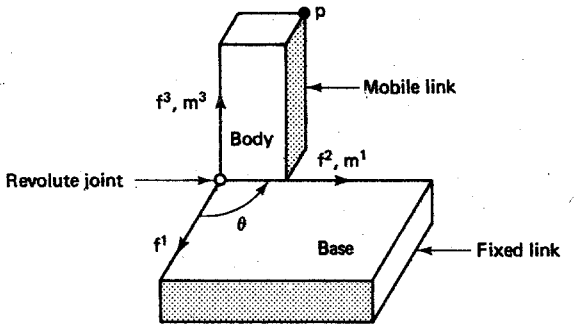


Figure 2-5 Single-axis robot.

tively. Note that m^2 is not shown in Fig. 2-5, since it is obscured by the body in the position shown, which corresponds to $\theta = \pi/2$. Since p is a point attached to the body, the coordinates of p with respect to the mobile coordinate frame M will remain fixed. However, the coordinates of p with respect to the fixed coordinate frame F will vary as the body is rotated. From Prop. 2-2-1, the two sets of coordinates of p are given by:

$$[p]^M = [p \cdot m^1, p \cdot m^2, p \cdot m^3]^T \tag{2-2-1}$$

$$[p]^F = [p \cdot f^1, p \cdot f^2, p \cdot f^3]^T \tag{2-2-2}$$

The problem then is to find the coordinates of p with respect to F, given the coordinates of p with respect to M. This is a coordinate transformation problem. The coordinate transformation problem has a particularly simple solution when the destination or reference coordinate frame F is orthonormal, as can be seen from the following result:

Proposition 2-2-2: Coordinate Transformations. Let $F = \{f^1, f^2, \ldots, f^n\}$ and $M = \{m^1, m^2, \ldots, m^n\}$ be coordinate frames for $\mathbf{R}^n$ with F orthonormal coordinate frames in $\mathbf{R}^n$ having the same origin. Next, let A be the $n \times n$ matrix defined by $A_{kj} \triangleq f^k \cdot m^j$ for $1 \leq k, j \leq n$. Then for each point p in $\mathbf{R}^n$:

$$[p]^F = A[p]^M$$

Proof. Since F is an orthonormal coordinate frame, it follows from Prop. 2-1-1, Prop. 2-2-1, and Def. 2-2-1 that for $1 \leq k \leq n$:

$$\begin{aligned} [p]_k^F &= p \cdot f^k \\ &= \left(\sum_{j=1}^{n} [p]_j^M m^j \right) \cdot f^k \\ &= \sum_{j=1}^{n} [p]_j^M (m^j \cdot f^k) \end{aligned}$$

$$= \sum_{j=1}^{n} (f^k \cdot m^j)[p]_j^M$$

$$= \sum_{j=1}^{n} A_{kj}[p]_j^M$$

Thus $[p]^F = A[p]^M$ where $A_{kj} = f^k \cdot m^j$ for $1 \leq k, j \leq n$.

We call the matrix A that maps or transforms mobile coordinates into fixed coordinates a *coordinate transformation matrix*. If a^j denotes the jth column of A, then, from Prop. 2-2-1 and Prop. 2-2-2, we find that $a^j = [m^j]^F$ for $1 \leq j \leq n$. Consequently, the jth column of the coordinate transformation matrix A is simply the coordinates of the jth vector of the source coordinate frame M with respect to the destination coordinate frame F.

Example 2-2-1: Coordinate Transformation

For the two coordinate frames pictured in Fig. 2-5, suppose the coordinates of the point p with respect to the mobile coordinate frame are measured and found to be $[p]^M = [0.6, 0.5, 1.4]^T$. What are the coordinates of p with respect to the fixed coordinate frame F with the body in the position shown?

Solution From Prop. 2-2-2, we have:

$$[p]^F = A[p]^M$$

$$= \begin{bmatrix} f^1 \cdot m^1 & f^1 \cdot m^2 & f^1 \cdot m^3 \\ f^2 \cdot m^1 & f^2 \cdot m^2 & f^2 \cdot m^3 \\ f^3 \cdot m^1 & f^3 \cdot m^2 & f^3 \cdot m^3 \end{bmatrix} \begin{bmatrix} 0.6 \\ 0.5 \\ 1.4 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.6 \\ 0.5 \\ 1.4 \end{bmatrix}$$

$$= [-0.5, 0.6, 1.4]^T$$

Note that this is consistent with the picture shown in Fig. 2-5.

If we have a coordinate transformation matrix which maps coordinates relative to one frame into coordinates relative to another frame and instead we want to move in the reverse direction, we need to find the inverse of the coordinate transformation matrix. When both the source and the destination coordinate frames are orthonormal, finding the inverse of the coordinate transformation matrix is particularly simple, as can be seen from the following useful result:

Proposition 2-2-3: Inverse Coordinate Transformation. Let F and M be two orthonormal coordinate frames in $\mathbf{R}^n$ having the same origin, and let A be the coordinate transformation matrix that maps M coordinates into F coordinates as in Prop. 2-2-2. Then the coordinate transformation matrix which maps F coordinates into M coordinates is given by A^{-1}, where:

$$A^{-1} = A^T$$

Proof. By multiplying both sides of the equation in Prop. 2-2-2 on the left by A^{-1}, we see that A^{-1} maps F coordinates into M coordinates. It remains to verify that the inverse is just the transpose. Let $1 \leq k, j \leq n$. Since both F and M are orthonormal, we can use Prop. 2-2-2 and the commutative property of the dot product as follows:

$$\begin{aligned}(A^{-1})_{kj} &= m^k \cdot f^j \\ &= f^j \cdot m^k \\ &= A_{jk} \\ &= (A^T)_{kj}\end{aligned}$$

Since the above equation holds identically for all $1 \leq k, j \leq n$, Prop. 2-2-3 then follows.

2-3 ROTATIONS

In order to specify the position and orientation of the mobile tool in terms of a coordinate frame attached to the fixed base, coordinate transformations involving both rotations and translations are required. We begin by investigating the representation of rotations.

2-3-1 Fundamental Rotations

If the mobile coordinate frame M is obtained from the fixed coordinate frame F by rotating M about one of the unit vectors of F, then the resulting coordinate transformation matrix is called a *fundamental rotation matrix*. In the space $\mathbf{R}^3$ there are three possibilities, as shown in Fig. 2-6.

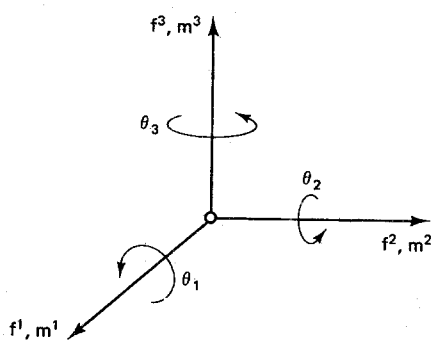


Figure 2-6 Fundamental rotations in $\mathbf{R}^3$.

As an illustration, suppose we rotate the mobile coordinate frame M about the f^1 axis of the fixed coordinate frame F. Let ϕ be the amount of rotation, measured in a right-handed sense, as shown in Fig. 2-7. Next let $R_1(\phi)$ be the resulting coordinate transformation matrix which maps mobile M coordinates into fixed F coordinates. Then, from Prop. 2-2-2:

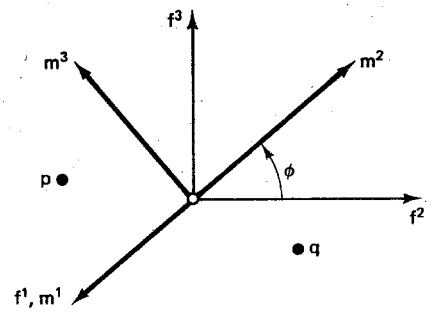


Figure 2-7 Rotation of M about f^1 by angle ϕ.

$$R_1(\phi) = \begin{bmatrix} f^1 \cdot m^1 & f^1 \cdot m^2 & f^1 \cdot m^3 \\ f^2 \cdot m^1 & f^2 \cdot m^2 & f^2 \cdot m^3 \\ f^3 \cdot m^1 & f^3 \cdot m^2 & f^3 \cdot m^3 \end{bmatrix} \tag{2-3-1}$$

Since we are rotating about the f^1 axis, it is clear from Fig. 2-7 that $f^1 = m^1$. Substituting this into Eq. (2-3-1) and recalling that $\{f^1, f^2, f^3\}$ and $\{m^1, m^2, m^3\}$ are orthonormal sets, it follows that the fundamental rotation matrix $R_1(\phi)$ simplifies to:

$$R_1(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & f^2 \cdot m^2 & f^2 \cdot m^3 \\ 0 & f^3 \cdot m^2 & f^3 \cdot m^3 \end{bmatrix} \tag{2-3-2}$$

Next note from Fig. 2-7 that the vectors $\{f^2, f^3, m^2, m^3\}$ all lie in a plane orthogonal to $f^1 = m^1$. Recalling Prop. 2-1-2, the dot product of two vectors is proportional to the cosine of the angle between the vectors. When the vectors are normal (unit length), the dot product is in fact equal to the cosine of the angle between the vectors. From Fig. 2-7 it is evident that the angle from f^2 to m^2 is ϕ, as is the angle from f^3 to m^3. Consequently the diagonal elements in the 2×2 submatrix in Eq. (2-3-2) are equal to $\cos \phi$.

In order to represent the off-diagonal elements in terms of ϕ, note from Fig. 2-7 that the angle from f^2 to m^3 is $\pi/2 + \phi$. Similarly, the angle from f^3 to m^2 is $-\pi/2 + \phi$. Using the trigonometric identity for the cosine of the sum (Appendix 1) and Prop. 2-1-2, the general form for the *first fundamental rotation matrix* then reduces to:

$$R_1(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \tag{2-3-3}$$

A similar analysis can be used to derive expressions for the *second* and the *third fundamental rotation matrices*. In particular, if $R_2(\phi)$ and $R_3(\phi)$ represent rotations by ϕ about the second and third unit vectors of the fixed coordinate frame F, then:

$$R_2(\phi) = \begin{bmatrix} \cos \phi & 0 & \sin \phi \\ 0 & 1 & 0 \\ -\sin \phi & 0 & \cos \phi \end{bmatrix} \tag{2-3-4}$$

$$R_3(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2-3-5}$$

There is a simple, consistent pattern to the structure of the three fundamental rotation matrices. The kth row and the kth column of $R_k(\phi)$ are identical to the kth row and the kth column of the identity matrix I. In the remaining 2×2 submatrix, the diagonal terms are $\cos\phi$, while the off-diagonal terms are $\pm\sin\phi$, where ϕ is the angle of rotation. The sign of the off-diagonal term above the diagonal is $(-1)^k$ for the kth fundamental rotation matrix.

Example 2-3-1: Rotation

Refer to Fig. 2-7, where the mobile coordinate frame M is rotated about the f^1 axis of the fixed coordinate frame F. Let $\phi = \pi/3$ radians be the amount of rotation. Suppose p is a point whose coordinates in the mobile M frame are $[p]^M = [2, 0, 3]^T$. What are the coordinates of p in the fixed coordinate frame F?

Solution Applying Prop. 2-2-2:

$$[p]^F = R_1\left(\frac{\pi}{3}\right)[p]^M$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5 & -0.866 \\ 0 & 0.866 & 0.5 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 3 \end{bmatrix}$$

$$= [2 \quad -2.598 \quad 1.5]^T$$

Next, suppose q is a point whose coordinates in the fixed coordinate frame F are given by $[q]^F = [3, 4, 0]^T$. What are the coordinates of q with respect to the mobile coordinate frame M?

Solution Since the two coordinate frames are orthonormal and have the same origin, the inverse of the coordinate transformation matrix is just the transpose. Applying Prop. 2-2-2 and Prop. 2-2-3:

$$[q]^M = R_1^{-1}\left(\frac{\pi}{3}\right)[q]^F$$

$$= R_1^T\left(\frac{\pi}{3}\right)[q]^F$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0.866 \\ 0 & -0.866 & 0.5 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \\ 0 \end{bmatrix}$$

$$= [3 \quad 2 \quad -3.464]^T$$

Note that both results are consistent with the picture shown in Fig. 2-7.

2-3-2 Composite Rotations

When a number of fundamental rotation matrices are multiplied together, the product matrix represents a sequence of rotations about the unit vectors. We refer to multiple rotations of this form as *composite rotations*. Using composite rotations, we can establish an arbitrary orientation for the robotic tool. Consider the sketch of a robotic tool shown in Fig. 2-8. Here the mobile coordinate frame $M = \{m^1, m^2, m^3\}$ rotates with the tool, while the fixed coordinate frame $F = \{f^1, f^2, f^3\}$ is stationary at the end of the forearm. The three fundamental rotations correspond to yaw, pitch, and roll. Recall from Chap. 1 that yaw is a rotation about the f^1 axis, pitch is a rotation about the f^2 axis, and roll is a rotation about the f^3 axis.

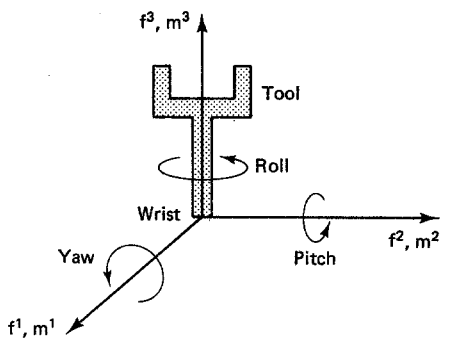


Figure 2-8 Yaw, pitch, and roll of tool.

Each fundamental rotation is represented by a matrix. However, the operation of matrix multiplication is not commutative, because for two arbitrary matrices A and B, it is *not* true that $AB = BA$. Consequently, the *order* in which the fundamental rotations are performed does make a difference in the resulting composite rotation. Furthermore, once one rotation has been performed, the axes of the two coordinate frames are no longer coincident. At this point subsequent rotations of the tool could be performed about the unit vectors of either the fixed coordinate frame F or the rotated coordinate frame M. In order to resolve the ambiguities as to how a composite rotation matrix should be constructed, the following algorithm is used:

Algorithm 2-3-1: Composite Rotations

1. Initialize the rotation matrix to $R = I$, which corresponds to the orthonormal coordinate frames F and M being coincident.
2. If the mobile coordinate frame M is to be rotated by an amount ϕ about the kth unit vector of the fixed coordinate frame F, then premultiply R by $R_k(\phi)$.
3. If the mobile coordinate frame M is to be rotated by an amount ϕ about its own kth unit vector, then postmultiply R by $R_k(\phi)$.
4. If there are more fundamental rotations to be performed, go to step [2]; else, stop. The resulting composite rotation matrix R maps mobile M coordinates into fixed F coordinates.

Thus composite rotation matrices are built up in steps starting with the identity matrix which corresponds to no rotation at all. Rotations of frame M about the unit vectors of frame F are represented by *premultiplication* (multiplication on the left) by the appropriate fundamental rotation matrix. Similarly, rotations of frame M about one of its own unit vectors are represented by *postmultiplication* (multiplication on the right) by the appropriate fundamental rotation matrix.

Since a convention has been adopted for the order of the yaw, pitch, and roll operations, a general expression for the composite yaw-pitch-roll transformation matrix can be obtained. Suppose that the yaw-pitch-roll angles are represented by a vector θ in $\mathbf{R}^3$. For notational convenience, let $S_k \triangleq \sin \theta_k$ and $C_k \triangleq \cos \theta_k$. We then have the following result, which summarizes the composite yaw-pitch-roll tool transformation:

Proposition 2-3-1: Yaw-Pitch-Roll Transformation. Let YPR (θ) represent the composite rotation matrix obtained by rotating a mobile coordinate frame $M = \{m^1, m^2, m^3\}$ first about unit vector f^1 with a yaw of θ_1, then about the unit vector f^2 with a pitch of θ_2, and finally about unit vector f^3 with a roll of θ_3. The resulting composite *yaw-pitch-roll matrix* YPR (θ) maps M coordinates into F coordinates and is given by:

$$\text{YPR}(\theta) = \begin{bmatrix} C_2C_3 & S_1S_2C_3 - C_1S_3 & C_1S_2C_3 + S_1S_3 \\ C_2S_3 & S_1S_2S_3 + C_1C_3 & C_1S_2S_3 - S_1C_3 \\ -S_2 & S_1C_2 & C_1C_2 \end{bmatrix}$$

Proof. This is verified by simply applying Algorithm 2-3-1 and using the expressions for the three fundamental rotation matrices. Using the notational shorthand $C_k = \cos \theta_k$ and $S_k = \sin \theta_k$ and Algorithm 2-3-1, we have:

$$\begin{aligned} \text{YPR}(\theta) &= R_3(\theta_3)R_2(\theta_2)R_1(\theta_1) \\ &= \begin{bmatrix} C_3 & -S_3 & 0 \\ S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & 0 & S_2 \\ 0 & 1 & 0 \\ -S_2 & 0 & C_2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_1 & -S_1 \\ 0 & S_1 & C_1 \end{bmatrix} \\ &= \begin{bmatrix} C_3 & -S_3 & 0 \\ S_3 & C_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & S_1S_2 & C_1S_2 \\ 0 & C_1 & -S_1 \\ -S_2 & S_1C_2 & C_1C_2 \end{bmatrix} \\ &= \begin{bmatrix} C_2C_3 & S_1S_2C_3 - C_1S_3 & C_1S_2C_3 + S_1S_3 \\ C_2S_3 & S_1S_2S_3 + C_1C_3 & C_1S_2S_3 - S_1C_3 \\ -S_2 & S_1C_2 & C_1C_2 \end{bmatrix} \end{aligned}$$

Example 2-3-2: Yaw-Pitch-Roll

Suppose we rotate the tool in Fig. 2-8 about the fixed axes starting with a yaw of $\pi/2$, followed by a pitch of $-\pi/2$ and, finally, a roll of $\pi/2$. What is the resulting composite rotation matrix?

Solution Applying Prop. 2-3-1:

$$R = R_3\left(\frac{\pi}{2}\right)R_2\left(\frac{-\pi}{2}\right)R_1\left(\frac{\pi}{2}\right)$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Suppose the point p at the tool tip has mobile coordinates $[p]^M = [0, 0, 0.6]^T$. Find $[p]^F$ following the yaw-pitch-roll transformation.

Solution Using Prop. 2-3-1:

$$[p]^F = R[p]^M$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0.6 \end{bmatrix}$$

$$= [0.6, 0, 0]^T$$

Exercise 2-3-1: Yaw-Pitch-Roll. Verify that Example 2-3-2 is correct by sketching the tool position and the coordinate frames after each fundamental rotation is performed.

Exercise 2-3-2: Roll-Pitch-Yaw. An alternative way to define the yaw-pitch-roll transformation is to perform the rotations about unit vectors of the mobile frame M and in the reverse order: roll, pitch, yaw. This tends to be easier to visualize, particularly when the angles are arbitrary. Show that the resulting composite rotation matrix RPY (θ), is the same as in Prop. 2-3-1. Is the route taken by the tool the same?

Exercise 2-3-3: Fundamental Rotations. Verify that the general yaw-pitch-roll transformation YPR (θ) includes the three fundamental rotations $\{R_1, R_2, R_3\}$ as special cases.

A general rotation of a mobile frame M with respect to a fixed frame F can always be decomposed into a sequence of three fundamental rotations as in Prop. 2-3-1. It is also possible to represent a general rotation as a *single* rotation by an angle ϕ about an *arbitrary* unit vector u as shown in Fig. 2-9. This is called the *equivalent angle-axis* representation. For notational convenience, let $C\phi \triangleq \cos\phi$, $S\phi \triangleq \sin\phi$, and $V\phi \triangleq 1 - \cos\phi$. The equivalent angle-axis rotation can then be represented as follows (Fu et al., 1987):

Proposition 2-3-2: Equivalent Angle-Axis. Let F and M be two orthonormal coordinate frames in $\mathbf{R}^3$ with M initially coincident with F. Let u be a unit vector, and suppose M is rotated about u by an angle ϕ in the right-handed sense. Then the *equivalent angle-axis rotation matrix* $R(\phi, u)$ which maps M coordinates into F coordinates is:

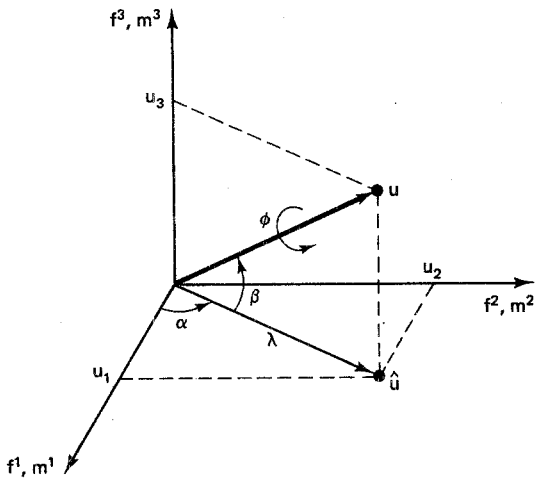


Figure 2-9 Equivalent angle-axis rotation.

$$R(\phi, u) = \begin{bmatrix} u_1^2 V\phi + C\phi & u_1u_2 V\phi - u_3 S\phi & u_1u_3 V\phi + u_2 S\phi \\ u_1u_2 V\phi + u_3 S\phi & u_2^2 V\phi + C\phi & u_2u_3 V\phi - u_1 S\phi \\ u_1u_3 V\phi - u_2 S\phi & u_2u_3 V\phi + u_1 S\phi & u_3^2 V\phi + C\phi \end{bmatrix}$$

Proof. Let $\hat{u} = [u_1, u_2, 0]^T$ denote the orthogonal projection of u onto the f^1f^2 plane as shown in Fig. 2-9. Then:

$$\begin{aligned} \lambda &= \|\hat{u}\| \\ &= (u_1^2 + u_2^2)^{1/2} \end{aligned}$$

Next, let α be the angle from f^1 to $\hat{u}$, and let β be the angle from $\hat{u}$ to u as shown in Fig. 2-9. Then, since $\|u\| = 1$, it follows that:

$$C\alpha = \frac{u_1}{\lambda}$$

$$S\alpha = \frac{u_2}{\lambda}$$

$$C\beta = \lambda$$

$$S\beta = u_3$$

We can express the equivalent angle-axis rotation matrix as a composition of fundamental rotation matrices. First, we rotate the mobile frame M about the f^3 axis by an angle of $-\alpha$. Thus:

$$R = R_3(-\alpha)$$

This puts the unit vector u in the f^1f^3 plane. Next we rotate frame M about the f^2 axis by an angle of β. Since the rotation is about an axis of the fixed frame, we premultiply to obtain the composite rotation matrix:

$$R = R_2(\beta)R_3(-\alpha)$$

This aligns unit vector u with the f^1 axis. At this point we perform the desired rotation by an angle ϕ about the unit vector $u = f^1$. Thus the composite rotation matrix is:

$$R = R_1(\phi)R_2(\beta)R_3(-\alpha)$$

Next we reverse the process to restore u to its original position. First we rotate frame M about f^2 by an angle of $-\beta$. This yields a composite rotation matrix of:

$$R = R_2(-\beta)R_1(\phi)R_2(\beta)R_3(-\alpha)$$

The last step is to rotate frame M about f^3 by an angle of α. This restores u to its original position and yields a final composite rotation matrix of:

$$R(\phi, u) = R_3(\alpha)R_2(-\beta)R_1(\phi)R_2(\beta)R_3(-\alpha)$$

It remains to substitute the expressions for the fundamental rotation matrices from Eqs. (2-3-3) through (2-3-5) into the expression for $R(\phi, u)$ and to perform the matrix multiplication. The procedure is tedious but straightforward, and the result after simplification using $C\alpha C\beta = u_1$, $S\alpha C\beta = u_2$, and $\lambda^2 + u_3^2 = 1$ is as given in Prop. 2-3-2.

The three fundamental rotation matrices represent important special cases of the equivalent angle-axis rotation matrix, as can be seen from the following exercise:

Exercise 2-3-4: Fundamental Rotations. Let $R(\phi, u)$ be the equivalent angle-axis rotation matrix and let $R_k(\phi)$ be the kth fundamental rotation matrix. Verify that:

$$R(\phi, i^k) = R_k(\phi) \qquad 1 \leq k \leq 3$$

The equivalent angle-axis representation for a general rotation can also be inverted (Paul, 1982). For example, given a rotation matrix R, suppose we want to find the axis u and angle ϕ. The sum of the diagonal terms of a square matrix is called the *trace* of the matrix. If we compute the trace of the matrix $R(\phi, u)$ in Prop. 2-3-2, then after simplification using trigonometric identities and $\|u\| = 1$, the result is trace $[R(\phi, u)] = 1 + 2\cos\phi$. Thus the angle part of the equivalent angle-axis representation is:

$$\phi = \arccos\left[\frac{\text{trace } (R) - 1}{2}\right] \tag{2-3-6}$$

Once the angle of rotation is identified over the range $(0, \pi)$, the axis or unit vector about which the rotation is performed can then be determined. In particular, if we form differences between off-diagonal terms of $R(\phi, u)$ in Prop. 2-3-2, we can isolate the components of u as follows:

$$u = \frac{1}{2\sin\phi}\begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix} \tag{2-3-7}$$

Note that u is well defined for $0 < \phi < \pi$, but at $\phi = 0$ and at $\phi = \pi$ the expressions for the components of u in Eq. (2-3-7) reduce to 0/0. Thus Eq. (2-3-7) is not well defined numerically for small values of ϕ or as ϕ approaches π. For a further discussion of these cases, see Paul (1982).

2-4 HOMOGENEOUS COORDINATES

Pure rotations are sufficient to characterize the orientation of a robotic tool. However, to characterize the position of the tool relative to a coordinate frame attached to the robot base, we must use another type of transformation, a translation. Translations are qualitatively different from rotations in one important respect. With rotations, the origin of the transformed coordinate frame is the same as the origin of the original coordinate frame. This invariance of the origin is characteristic of *linear* operations in general, and it is this feature that allows us to represent rotations in three-dimensional space with 3×3 matrices. However, the origin of a translated coordinate frame is not the same as the origin of the original coordinate frame. Consequently, it is not possible to represent a translation in $\mathbf{R}^3$ with a 3×3 matrix.

2-4-1 Homogeneous Coordinate Frames

Instead we must move to a higher-dimensional space, the four-dimensional space of homogeneous coordinates. We define the homogeneous coordinates of a point as follows:

Definition 2-4-1: Homogeneous Coordinates. Let q be a point in $\mathbf{R}^3$, and let F be an orthonormal coordinate frame for $\mathbf{R}^3$. If σ is any nonzero *scale factor*, then the *homogeneous coordinates* of q with respect to F are denoted $[q]^F$ and defined:

$$[q]^F \triangleq \sigma[q_1, q_2, q_3, 1]^T$$

Thus the homogeneous coordinates of a point q in $\mathbf{R}^3$ are represented by a vector $[q]^F$ in four-dimensional space $\mathbf{R}^4$. The fourth component σ is a nonzero scale factor. To recover the original physical three-dimensional vector from its four-dimensional homogeneous coordinates, we can use $q = H_\sigma[q]^F$, where H_σ is a 3×4 *homogeneous coordinate conversion matrix*, defined as follows:

$$H_\sigma \triangleq \frac{1}{\sigma}[I, 0] \tag{2-4-1}$$

Note that the homogeneous coordinates $[q]^F$ are *not unique*, because *any* scale factor $\sigma \neq 0$ will yield the *same* three-dimensional physical vector q. In robotics, we normally use a scale factor of $\sigma = 1$ for convenience. In this case, four-dimensional homogeneous coordinates are obtained from three-dimensional physical coordinates by simply augmenting the vector with a unit fourth component. Similarly, three-dimensional physical coordinates are recovered from four-dimensional homogeneous coordinates by merely dropping the unit fourth component.

If a physical point in three-dimensional space is expressed in terms of its homogeneous coordinates and we want to change from one coordinate frame to another, we use a 4×4 *homogeneous transformation matrix*. In general, a homogeneous transformation matrix T can be partitioned into four separate submatrices as follows:

$$T \triangleq \begin{array}{c} \begin{array}{cc} \text{rotation} & \text{translation} \end{array} \\ \left[\begin{array}{c|c} R & p \\ \hline \eta^T & \sigma \end{array}\right] \\ \begin{array}{cc} \text{perspective} & \text{scale} \end{array} \end{array} \tag{2-4-2}$$

Here the 3×3 submatrix R in the upper left corner of T is a *rotation matrix*. It represents the orientation of the mobile coordinate frame relative to the fixed reference frame. The 3×1 column vector p in the upper right corner of T is a *translation vector*. It represents the position of the origin of the mobile coordinate frame relative to the fixed reference frame. The scalar σ in the lower right corner of T is a nonzero *scale factor* which is typically set to unity. Finally, the 1×3 row vector η^T in the lower left corner of T is a *perspective vector*. For the time being, this vector will always be $\eta = 0$. Later, when we examine the use of an overhead camera in Chap. 8, nonzero perspectives will be used. In terms of a robotic arm, p specifies the position of the tool tip, R specifies the orientation of the tool, η specifies a point of perspective for visual sensing with a camera, and σ is typically set to unity for standard scaling.

2-4-2 Translations and Rotations

The fundamental operations of translation and rotation can each be regarded as important special cases of the general 4×4 homogeneous transformation matrix. To see this, first consider the question of rotation. Suppose F and M are two orthonormal coordinate frames that are initially coincident. If we rotate M by an amount ϕ about the kth unit vector of F, then in terms of homogeneous coordinates this operation can be represented by a 4×4 matrix denoted Rot (ϕ, k), where:

$$\text{Rot}(\phi, k) \triangleq \left[\begin{array}{ccc|c} & & & 0 \\ & R_k(\phi) & & 0 \\ & & & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \qquad 1 \leq k \leq 3 \tag{2-4-3}$$

Here $R_k(\phi)$ is simply the kth fundamental rotation matrix introduced in Sec. 2-3. We refer to Rot (ϕ, k) as the kth *fundamental homogeneous rotation matrix*. Composite homogeneous rotation matrices are then built up from the fundamental homogeneous rotation matrices in a manner analogous to the method summarized in Algorithm 2-3-1. Note that the translation vector p in each case has been set to zero.

Example 2-4-1: Homogeneous Rotation

Suppose that $[q]^M = [0, 0, 10, 1]^T$ represents the homogeneous coordinates of a point located 10 units along the third vector of a mobile coordinate frame M. Suppose that initially M is coincident with a fixed coordinate frame F. If we rotate the mobile M frame by $\pi/4$ radians about the first unit vector of F, then the resulting homogeneous coordinate transformation matrix is:

$$\text{Rot}\left(\frac{\pi}{4}, 1\right) = \left[\begin{array}{ccc:c} & R_1\left(\frac{\pi}{4}\right) & & \begin{matrix}0\\0\\0\end{matrix} \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right]$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} & 0 \\ 0 & \sin\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.707 & -0.707 & 0 \\ 0 & 0.707 & 0.707 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Hence the homogeneous coordinates of the point q in the fixed coordinate frame F following the rotation are:

$$[q]^F = \text{Rot}\left(\frac{\pi}{4}, 1\right)[q]^M$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.707 & -0.707 & 0 \\ 0 & 0.707 & 0.707 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 10 \\ 1 \end{bmatrix}$$

$$= [0 \quad -7.07 \quad 7.07 \quad 1]^T$$

Finally, the physical coordinates of the point q in the fixed coordinate frame F are $q = [0, -7.07, 7.07]^T$.

The power of homogeneous coordinates lies in the fact that matrices can also be used to represent translations. For example, let F and M be two initially coincident orthonormal coordinate frames, and suppose we want to translate the origin of the mobile coordinate frame M by an amount p_k along the kth unit vector of F for $1 \leq k \leq 3$. Then in terms of homogeneous coordinates, this operation can be repre-

sented by a 4×4 matrix denoted Tran (p), where:

$$\text{Tran}\,(p) \triangleq \left[\begin{array}{ccc:c} 1 & 0 & 0 & p_1 \\ 0 & 1 & 0 & p_2 \\ 0 & 0 & 1 & p_3 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{2-4-4}$$

We refer to Tran (p) as the *fundamental homogeneous translation matrix*. Note that the translation vector is p, while the 3×3 rotation matrix R has been set to the identity matrix I.

Example 2-4-2: Homogeneous Translation

Suppose the homogeneous coordinates of a point are $[q]^M = [0, 0, 10, 1]^T$, as in Example 2-4-1. However, this time suppose we translate the mobile M coordinate frame relative to the fixed F coordinate frame by 5 units along the f^1 axis and -3 units along the f^2 axis. Then the homogeneous transformation matrix which represents this operation is:

$$\text{Tran}\,(p) = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

It follows that the homogeneous coordinates of point q relative to the reference frame F following the translation are given by:

$$\begin{aligned} [q]^F &= \text{Tran}\,(p)\,[q]^M \\ &= \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & -3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 10 \\ 1 \end{bmatrix} \\ &= [5 \quad -3 \quad 10 \quad 1]^T \end{aligned}$$

Finally, the physical coordinates of the point q in the fixed F coordinate frame following the translation are $q = [5, -3, 10]^T$.

Exercise 2-4-1: Inverse Translation. Verify that the inverse of the fundamental homogeneous translation matrix Tran (p) always exists and is given by:

$$\text{Tran}^{-1}\,(p) = \text{Tran}\,(-p)$$

Exercise 2-4-2: Inverse Rotation. Verify that the inverse of the fundamental homogeneous rotation matrix Rot (ϕ, k) always exists and is given by:

$$\text{Rot}^{-1}\,(\phi, k) = \text{Rot}\,(-\phi, k) = \text{Rot}^T\,(\phi, k) \qquad 1 \leq k \leq 3$$

2-4-3 Composite Homogeneous Transformations

In the general case, a homogeneous transformation matrix will represent both a rotation and a translation of the mobile frame with respect to the fixed frame. A sequence of individual rotations and translations can be represented as a product of fundamental homogeneous transformation matrices. However, because matrix multiplication is not a commutative operation, the order in which the rotations and translations are to be performed is important. Furthermore, the mobile coordinate frame can be rotated or translated about the unit vectors of the fixed frame or about its own unit vectors. The effects of these different operations on the composite transformation matrix are summarized in the following algorithm:

Algorithm 2-4-1: Composite Homogeneous Transformations

1. Initialize the transformation matrix to $T = I$, which corresponds to the orthonormal coordinate frames F and M being coincident.
2. Represent rotations and translations using separate homogeneous transformation matrices.
3. Represent composite rotations as separate fundamental homogeneous rotation matrices.
4. If the mobile coordinate frame M is to be rotated about or translated along a unit vector of the fixed coordinate frame F, premultiply the homogeneous transformation matrix T by the appropriate fundamental homogeneous rotation or translation matrix.
5. If the mobile coordinate frame M is to be rotated about or translated along one of its own unit vectors, postmultiply the homogeneous transformation matrix T by the appropriate fundamental homogeneous rotation or translation matrix.
6. If there are more fundamental rotations or translations to be performed, go to step 4; otherwise, stop. The resulting composite homogeneous transformation matrix T maps mobile M coordinates into fixed F coordinates.

Thus composite homogeneous transformation matrices are built up in steps starting with the identity matrix, which corresponds to coincident mobile and fixed frames. Rotations and translations of frame M along the unit vectors of frame F are represented by *premultiplication* (multiplication on the left) by the appropriate fundamental homogeneous transformation matrix. Similarly, rotations and translations of frame M along one of its own unit vectors are represented by *postmultiplication* (multiplication on the right) by the appropriate fundamental homogeneous transformation matrix.

Example 2-4-3: Composite Homogeneous Transformation

Let $F = \{f^1, f^2, f^3\}$ and $M = \{m^1, m^2, m^3\}$ be two initially coincident fixed and mobile orthonormal coordinate frames, respectively. Suppose we translate M along f^2 by 3 units and then rotate M about f^3 by π radians. Find $[m^1]^F$ after the composite transformation.

Solution Using Algorithm 2-4-1:

$$
\begin{aligned}
[m^1]^F &= T[m^1]^M \\
&= \text{Rot}\,(\pi, 3)\ \text{Tran}\,(3i^2)\ [m^1]^M \\
&= \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\
&= [-1 \quad -3 \quad 0 \quad 1]^T
\end{aligned}
$$

Thus, in terms of physical coordinates, $[m^1]^F = [-1, -3, 0]^T$. This is consistent with the diagram of the two coordinate frames displayed in Fig. 2-10.

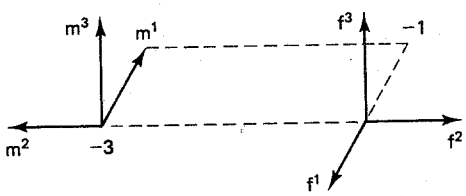


Figure 2-10 Coordinate frames for Example 2-4-3.

Example 2-4-4: Composite Homogeneous Transformation

Suppose we repeat the transformation of Example 2-4-3 but this time with the order of the operations reversed. First we rotate M by π radians about f^3, and then we translate the rotated M by 3 units along f^2. Find $[m^1]^F$ following the composite transformation.

Solution Using Algorithm 2-4-1:

$$
\begin{aligned}
[m^1]^F &= T[m^1]^M \\
&= \text{Tran}\,(3i^2)\ \text{Rot}\,(\pi, 3)\ [m^1]^M \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \\
&= [-1 \quad 3 \quad 0 \quad 1]^T
\end{aligned}
$$

Thus, in terms of physical coordinates, $[m^1]^F = [-1, 3, 0]^T$. Again, this is consistent with the diagram of the two coordinate frames displayed in Fig. 2-11. Note that the

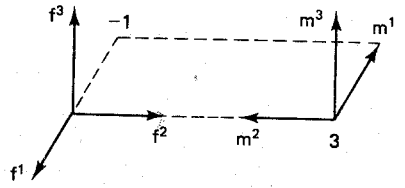


Figure 2-11 Coordinate frames for Example 2-4-4.

final position of the mobile frame M in Fig. 2-11 differs from that in Fig. 2-10. Hence the *order* in which the fundamental transformations are performed is important.

The geometric interpretation of the homogeneous transformation matrix T is that it provides the position and the orientation of a mobile coordinate frame M with respect to a fixed reference frame F. In the analysis of robotic manipulators, we will sometimes also want to express the position and orientation of the fixed frame F in terms of the mobile frame M. Consequently, we need to find the inverse of the homogeneous transformation matrix T. Because T is no longer a pure rotation between orthonormal frames, the inverse of T is not simply the transpose. However, when the perspective vector is $\eta = 0$ and the scale factor is $\sigma = 1$, the inverse of T does have a particularly simple form, as can be seen from the following useful result:

Proposition 2-4-1: Inverse Homogeneous Transformation. Let T be a homogeneous coordinate transformation matrix with rotation R and translation p between two orthonormal coordinate frames. If $\eta = 0$ and $\sigma = 1$, then the inverse of T is:

$$T^{-1} = \left[\begin{array}{ccc:c} & R^T & & -R^T p \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right]$$

Proof. Since we have a candidate for T^{-1}, it is simply a matter of evaluating it to see whether it satisfies the definition of the inverse. The two matrices T and T^{-1} are partitioned conformably for multiplication, hence the submatrices can be treated in a manner similar to scalars. Thus:

$$T^{-1}T = \left[\begin{array}{ccc:c} & R^T & & -R^T p \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right]\left[\begin{array}{ccc:c} & R & & p \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right]$$

$$= \left[\begin{array}{ccc:c} & & & 0 \\ & R^T R & & 0 \\ & & & 0 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right]$$

But since R is a pure rotation between orthonormal coordinate frames, it follows from Prop. 2-2-3 that $R^T = R^{-1}$. Thus $T^{-1}T = I$, where T^{-1} is as in Prop. 2-4-1. A similar analysis reveals that $TT^{-1} = I$. Hence the inverse of T is as given in Prop. 2-4-1.

In view of Prop. 2-4-1, we can just as easily transform coordinates from the base to the tool tip as from the tool tip back to the base. There is no need to perform computationally expensive and potentially inaccurate numerical inverses.

Example 2-4-5: Inverse Homogeneous Transformation

Let $F = \{f^1, f^2, f^3\}$ and $M = \{m^1, m^2, m^3\}$ be fixed and mobile orthonormal coordinate frames as shown in Fig. 2-12. Here the homogeneous coordinate transformation matrix which maps mobile M coordinates into fixed F coordinates is given by:

$$T = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Find the homogeneous coordinate transformation which maps fixed F coordinates into mobile M coordinates, and use it to find $[f^2]^M$.

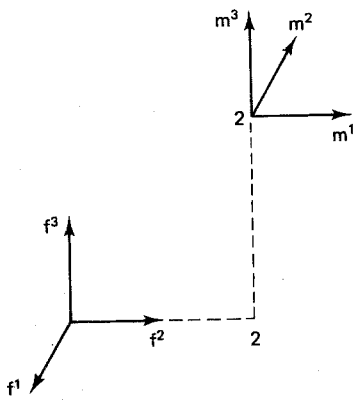


Figure 2-12 Coordinate frames for Example 2-4-5.

Solution Using Prop. 2-4-1, we have:

$$[f^2]^M = T^{-1}[f^2]^F$$

$$= \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 & -2 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$= [-1 \quad 0 \quad -2 \quad 1]^T$$

Thus, in terms of physical coordinates, $[f^2]^M = [-1, 0, -2]^T$. This is consistent with Fig. 2-12.

2-4-4 Screw Transformations

Certain composite homogeneous transformations arise repeatedly in robotic applications. One is a linear displacement along an axis combined with an angular displacement about the same axis. This type of motion corresponds to a threading or an unthreading operation and is therefore referred to as a *screw transformation.*

Definition 2-4-2: Screw Transformation. Let F and M be initially coincident fixed and mobile orthonormal coordinate frames, respectively. If M is translated along the kth unit vector of F by a displacement of λ and rotated about the kth unit vector of F by an angle of ϕ, the resulting composite homogeneous coordinate transformation matrix is called the *kth fundamental screw transformation matrix.* It is denoted Screw (λ, ϕ, k) and can be expressed:

$$\text{Screw } (\lambda, \phi, k) \triangleq \text{Rot } (\phi, k) \text{ Tran } (\lambda i^k)$$

It is clear that the screw transformation matrix includes the fundamental homogeneous rotation matrices as special cases, because Screw $(0, \phi, k) = \text{Rot } (\phi, k)$ for $1 \leq k \leq 3$. Similarly, a translation along a unit vector is also a special case, because Screw $(\lambda, 0, k) = \text{Tran } (\lambda i^k)$. As with a physical screw, one can define a pitch associated with the thread. In the case of homogeneous transformations, the *screw pitch* is defined:

$$\rho \triangleq \frac{\phi}{2\pi\lambda} \text{ threads per unit length} \tag{2-4-5}$$

For a pure rotation, the linear displacement is $\lambda = 0$. Consequently, a *pure rotation* is a screw with *infinite* pitch. Similarly, for a pure translation along a unit vector, $\phi = 0$. Thus a *pure translation* is a screw with a *zero* pitch. Note that if ρ is positive, then the screw is *right-handed,* whereas if ρ is negative it is *left-handed.* Refer to Fig. 2-13 for a summary of the relationship between translations, rotations, and screws. Here each straight line through the origin corresponds to a screw with a specific pitch. Horizontal lines are pure translations, while vertical lines are pure rotations.

The formulation of a screw transformation in Def. 2-4-2 consists of a translation along f^k by a displacement of λ followed by a rotation about f^k by an angle ϕ. It

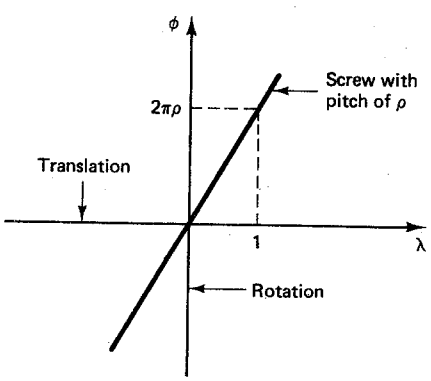


Figure 2-13 Translations, rotations, and screws.

could instead have been formulated the other way around as a rotation followed by a translation, as can be seen from the following exercise. This is a consequence of the fact that the rotation and translation operations are being performed along the *same* axis.

Exercise 2-4-3: Screw Transformation. Show that the fundamental rotation and translation matrices associated with the unit vectors commute. That is,

$$\text{Tran}\,(\lambda i^k)\ \text{Rot}\,(\phi, k) = \text{Rot}\,(\phi, k)\ \text{Tran}\,(\lambda i^k) \qquad 1 \le k \le 3$$

Exercise 2-4-4: Inverse Screw Transformation. Verify that the inverse of the screw transformation is again a screw transformation with:

$$\text{Screw}^{-1}\,(\lambda, \phi, k) = \text{Screw}\,(-\lambda, -\phi, k)$$

Example 2-4-6: Screw Transformation

Let $F = \{f^1, f^2, f^3\}$ and $M = \{m^1, m^2, m^3\}$ be initially coincident fixed and mobile orthonormal coordinate frames, respectively. Suppose we perform a screw transformation along axis f^2 translating by a distance of $\lambda = 3$ and rotating by an angle of $\pi/2$. Find $[m^3]^F$ following the screw transformation, and determine the pitch of the screw.

Solution From Def. 2-4-2, we have:

$$\begin{aligned}
[m^3]^F &= \text{Screw}\left(3, \frac{\pi}{2}, 2\right)[m^3]^M \\
&= \text{Tran}\,(3i^2)\ \text{Rot}\left(\frac{\pi}{2}, 2\right)[m^3]^M \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 3 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \\
&= [1 \quad 3 \quad 0 \quad 1]^T
\end{aligned}$$

Thus, in terms of physical coordinates, $[m^3]^F = [1, 3, 0]^T$. This is consistent with the diagram of the two coordinate frames displayed in Fig. 2-14. The pitch of the screw is $\rho = \frac{1}{12}$ threads per unit length.

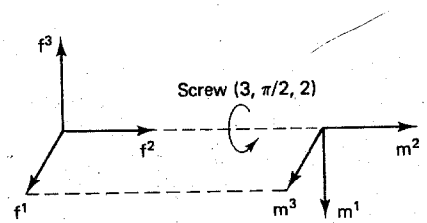


Figure 2-14 Coordinate frames for Example 2-4-6.

Exercise 2-4-5: General Screw Transformation. Let Scr (λ, ϕ, u) denote a screw with a linear displacement of λ and an angular displacement of ϕ about an arbitrary unit vector u. Find a formulation for Scr (λ, ϕ, u) and show that:

$$\text{Scr } (\lambda, \phi, i^k) = \text{Screw } (\lambda, \phi, k) \qquad 1 \leq k \leq 3$$

2-5 LINK COORDINATES

Recall that a robotic arm can be modeled as a chain of rigid links interconnected by either revolute or prismatic joints. The objective of this section is to systematically assign coordinate frames to each of those links. Once this is done, a general *arm equation* which represents the kinematic motion of the links of the manipulator can then be obtained. We begin by examining certain parameters associated with the physical design of a robotic arm.

2-5-1 Kinematic Parameters

Every adjacent pair of links is connected by either a revolute or a prismatic joint. The relative position and orientation of two successive links can be specified by two *joint parameters*, as shown in Fig. 2-15.

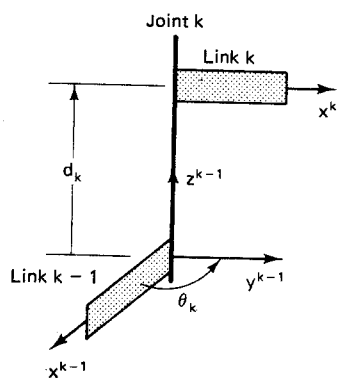


Figure 2-15 Joint angle θ and joint distance d.

Here joint k connects link $k-1$ to link k. The parameters associated with joint k are defined with respect to z^{k-1}, which is aligned with the axis of joint k. The first joint parameter, θ_k, is called the *joint angle*. It is the rotation about z^{k-1} needed to make axis x^{k-1} parallel with axis x^k. The second joint parameter, d_k, is called the *joint distance*. It is the translation along z^{k-1} needed to make axis x^{k-1} intersect with axis x^k. Thus θ_k is a rotation about the axis of joint k, while d_k is a translation along the axis of joint k. For each joint, it will always be the case that one of these parameters is fixed and the other is variable. The variable joint parameter depends on the type of joint, as indicated in Table 2-1. For a revolute joint, the joint angle θ_k is variable and the joint distance d_k is fixed. Thus the two links *rotate* relative to one

TABLE 2-1 KINEMATIC PARAMETERS

Arm Parameter	Symbol	Revolute Joint (R)	Prismatic Joint (P)
Joint angle	θ	Variable	Fixed
Joint distance	d	Fixed	Variable
Link length	a	Fixed	Fixed
Link twist angle	α	Fixed	Fixed

another about the axis of joint k. For a prismatic joint, the joint distance d_k is variable and the joint angle θ_k is fixed. In this case the two links *translate* relative to one another along the axis of joint k.

Just as there is a joint connecting adjacent links, there is a link between successive joints. The relative position and orientation of the axes of two successive joints can be specified by two *link parameters,* as shown in Fig. 2-16. Here link k connects joint k to joint $k + 1$. The parameters associated with link k are defined with respect to x^k, which is a common normal between the axes of joint k and joint $k + 1$. The first link parameter, a_k, is called the *link length*. It is the translation along x^k needed to make axis z^{k-1} intersect with axis z^k. The second link parameter, α_k, is called the *link twist angle*. It is the rotation about x^k needed to make axis z^{k-1} parallel with axis z^k.

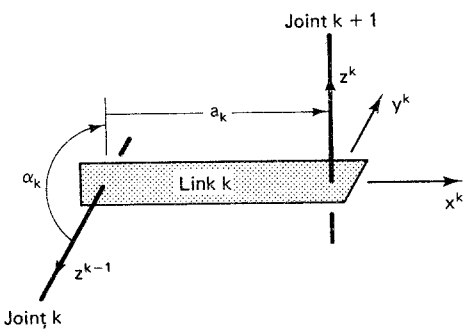


Figure 2-16 Link length a and link twist angle α.

Unlike the joint parameters, the two link parameters are always constant and are specified as part of the mechanical design. For industrial robots, the link twist angle is usually a multiple of $\pi/2$ radians. Sometimes the axes of joint k and joint $k + 1$ intersect, in which case the length of link k is zero. Links of length zero occur, for example, in robots with spherical wrists, where the last $n - 3$ axes intersect. More generally, robots can be designed to have many of the constant kinematic parameters equal to zero, in which case they are referred to as *kinematically simple* manipulators.

For an n-axis robotic manipulator, the $4n$ kinematic parameters constitute the *minimal* set needed to specify the kinematic configuration of the robot. For each axis, three of the parameters are fixed and depend on the mechanical design, while the fourth parameter is the joint variable, as indicated in Table 2-1. For a rectangular-coordinate robot the first three joint variables are all joint distances, whereas for an articulated-coordinate robot the first three joint variables are all joint

angles. Between these two extremes are cylindrical, spherical, and SCARA robots, whose first three joint variables are a combination of joint angles and joint distances.

2-5-2 Normal, Sliding, and Approach Vectors

By convention, the joints and links of a robotic arm are numbered outward starting with the fixed base, which is link 0, and ending with the tool, which is link n. For an n-axis robot there are $n + 1$ links interconnected by n joints, with joint k connecting link $k - 1$ to link k. In order to systematically assign coordinate frames to the links of an n-axis robotic arm, special attention must be paid to the last link—the tool, or end-effector. The orientation of the tool can be expressed in rectangular coordinates by a rotation matrix $R = [r^1, r^2, r^3]$, where the three columns of R correspond to the normal, sliding, and approach vectors, respectively, as shown in Fig. 2-17. The *approach vector* r^3 is aligned with the tool roll axis and points away from the wrist. Consequently, the approach vector specifies the direction in which the tool is pointing. The *sliding vector* r^2 is orthogonal to the approach vector and aligned with the open-close axis of the tool. Finally, the *normal vector* r^1 is orthogonal to the plane defined by the approach and sliding vectors and completes a right-handed orthonormal coordinate frame.

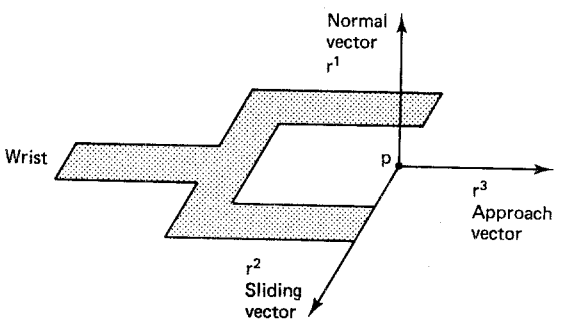


Figure 2-17 Normal, sliding, and approach vectors of the tool.

Recall that the yaw, pitch, and roll motions are rotations about the normal, sliding, and approach vectors, respectively. For the purpose of representing tool orientation, the origin of the $\{r^1, r^2, r^3\}$ frame can be taken either at the wrist, as in Chap. 1, or at the tool tip, as in Fig. 2-17. For the remainder of this text we will assume that the origin of the $\{r^1, r^2, r^3\}$ frame is placed at the tool tip, as in Fig. 2-17, because this simplifies the subsequent representation of tool-tip position.

2-5-3 The Denavit-Hartenberg (D-H) Representation

Denavit and Hartenberg (1955) proposed a systematic notation for assigning right-handed orthonormal coordinate frames, one to each link in an open kinematic chain of links. Once these link-attached coordinate frames are assigned, transformations between adjacent coordinate frames can then be represented by a single standard 4×4 homogeneous coordinate transformation matrix. To assign coordinate frames to the links of the robotic manipulator, let L_k be the frame associated with *link k*. That is:

کتابخانه دانشگاه علم و صنعت ایران
واحد اراک

$$L_k \triangleq \{x^k, y^k, z^k\} \qquad 0 \leq k \leq n \tag{2-5-1}$$

Coordinate frame L_k will be attached to the *distal end* of link k for $0 \leq k \leq n$. This puts the last coordinate frame, L_n, at the tool tip. The coordinate frames are assigned to the links using the the following procedure:

Algorithm 2-5-1: D-H Representation

0. Number the joints from 1 to n starting with the base and ending with the tool yaw, pitch, and roll, in that order.

1. Assign a right-handed orthonormal coordinate frame L_0 to the robot base, making sure that z^0 aligns with the axis of joint 1. Set $k = 1$.

2. Align z^k with the axis of joint $k + 1$.

3. Locate the origin of L_k at the intersection of the z^k and z^{k-1} axes. If they do not intersect, use the intersection of z^k with a common normal between z^k and z^{k-1}.

4. Select x^k to be orthogonal to both z^k and z^{k-1}. If z^k and z^{k-1} are parallel, point x^k away from z^{k-1}.

5. Select y^k to form a right-handed orthonormal coordinate frame L_k.

6. Set $k = k + 1$. If $k < n$, go to step 2; else, continue.

7. Set the origin of L_n at the tool tip. Align z^n with the approach vector, y^n with the sliding vector, and x^n with the normal vector of the tool. Set $k = 1$.

8. Locate point b^k at the intersection of the x^k and z^{k-1} axes. If they do not intersect, use the intersection of x^k with a common normal between x^k and z^{k-1}.

9. Compute θ_k as the angle of rotation from x^{k-1} to x^k measured about z^{k-1}.

10. Compute d_k as the distance from the origin of frame L_{k-1} to point b^k measured along z^{k-1}.

11. Compute a_k as the distance from point b^k to the origin of frame L_k measured along x^k.

12. Compute α_k as the angle of rotation from z^{k-1} to z^k measured about x^k.

13. Set $k = k + 1$. If $k \leq n$, go to step 8; else, stop.

For convenience, we will refer to Algorithm 2-5-1 as the *D-H algorithm*. Note that it is essentially a two-pass algorithm in terms of the manipulator. On the first pass (steps 0 through 7), a set of $n + 1$ right-handed orthonormal coordinate frames $\{L_0, L_1, \ldots, L_n\}$ is assigned, one to the distal end of each link. On the second pass (steps 8 through 13), the values for the kinematic parameters $\{\theta, d, a, \alpha\}$ are determined. Note that in step 8 axis x^k should always intersect with axis z^{k-1} when $k < n$. When $k = n$, x^k will intersect with z^{k-1} if the last joint is a tool roll joint.

2-5-4 An Example: Microbot Alpha II

As an example of an application of the D-H algorithm, consider the Alpha II robotic arm pictured in Fig. 2-18. This is a small table-top robotic arm manufactured by Microbot, Inc. It is a five-axis articulated-coordinate robotic manipulator that uses stepper motors for the joint actuators.

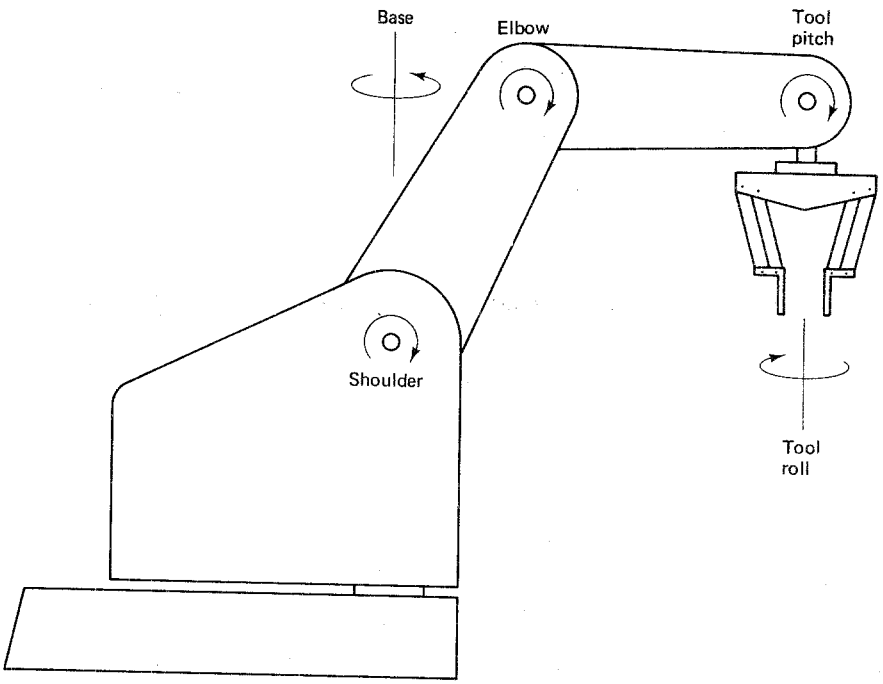


Figure 2-18 Alpha II robotic arm.

Applying steps 0 to 7 of the D-H algorithm, we get the link-coordinate diagram shown in Fig. 2-19. Here the dotted diagonal line between the origin of L_3 and the origin of L_4 indicates that the origins of these two coordinate frames coincide. They are drawn separated in order to make the diagram more clear. Note that the assignment of link coordinates dictated by the D-H algorithm is *not unique*. For example, the directions of any of the z axes could be reversed. Of course, this would also require reversing the corresponding y axes in order to preserve the right-handed nature of the orthonormal coordinate frames.

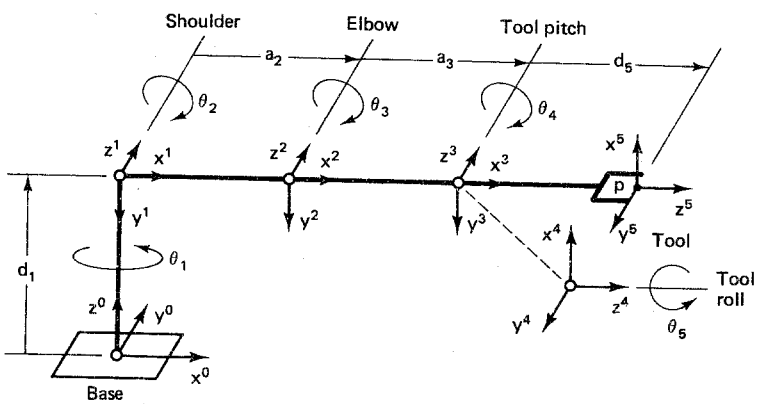


Figure 2-19 Link coordinates of the Alpha II robotic arm.

Next we apply steps 8 to 13 of the D-H algorithm starting with $k = 1$. Using Fig. 2-19, we get the set of kinematic parameters for the Alpha II robot shown in Table 2-2. Since the Alpha II is a five-axis articulated-coordinate robot, the vector of joint variables is the 5×1 vector θ. Note that 9 of the 15 fixed parameters in Table 2-2 are zero, which makes the Alpha II robot kinematically simple. If the origin of the base frame L_0 were to be moved up to coincide with the origin of frame L_1 (allowable under the D-H algorithm), then the kinematic parameters could be simplified still further with $d_1 = 0$.

TABLE 2-2 KINEMATIC PARAMETERS FOR THE ALPHA II ROBOT

Axis	θ	d	a	α	Home
1	θ_1	215.0 mm	0	$-\pi/2$	0
2	θ_2	0	177.8 mm	0	0
3	θ_3	0	177.8 mm	0	0
4	θ_4	0	0	$-\pi/2$	$-\pi/2$
5	θ_5	129.5 mm	0	0	0

The values listed in the last column of Table 2-2 for the joint variable θ correspond to the *soft home* position pictured in the link-coordinate diagram of Fig. 2-19. When assigning link-coordinates using the D-H algorithm, it is always easier to choose a soft home position that corresponds to the joint angles being multiples of $\pi/2$ radians even if the original picture of the robot is not so configured.

It is evident from inspection of Fig. 2-19 that the origins of the coordinate frames associated with the tool orientation $\{L_3, L_4\}$ coincide at the wrist. This is indicative of the fact that the Alpha II robot has a spherical wrist. However, it is not a general spherical wrist, because there is no tool yaw motion. The mechanical design that enables the Alpha II robot to achieve a pitch-roll spherical wrist is shown in Fig. 2-20.

The wrist design employs a set of three bevel gears $\{A, B, C\}$ in a "universal" joint configuration as shown in Fig. 2-20. Gears A and B are driven by separate stepper motors and cables, while gear C is attached to the tool roll axis. If gears A and B are driven in opposite directions at the same speed, the tool will exhibit a pure *roll* motion. However, if gears A and B are driven in the same direction at the same speed, the tool will exhibit a pure *pitch* motion with the tool tip moving "into" and

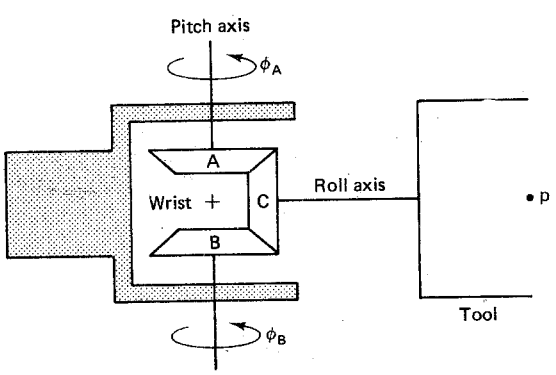


Figure 2-20 Pitch-roll spherical wrist using bevel gears.

"out of" the page. More complex motions involving simultaneous pitch and roll movement can also be achieved. It is clear that the pitch and roll axes intersect at the wrist.

2-6 THE ARM EQUATION

Once a set of link coordinates is assigned using the D-H algorithm, we can then transform from coordinate frame k to coordinate frame $k-1$ using a homogeneous coordinate transformation matrix. By multiplying several of these coordinate transformation matrices together, we arrive at a composite coordinate transformation matrix which transforms or maps tool coordinates into base coordinates. This composite homogeneous coordinate transformation matrix is called the *arm matrix*.

2-6-1 The Arm Matrix

There are four steps involved in constructing the homogeneous transformation matrix which maps frame k coordinates into frame $k-1$ coordinates. Each of these steps is associated with one of the four kinematic parameters summarized in Table 2-1. To determine the transformation matrix, we successively rotate and translate coordinate frame $k-1$ to render it coincident with coordinate frame k. Using steps 8 to 12 of the D-H algorithm, this involves the four fundamental operations summarized in Table 2-3.

TABLE 2-3 TRANSFERRING FRAME $k-1$ TO FRAME k

Operation	Description
1	Rotate L_{k-1} about z^{k-1} by θ_k.
2	Translate L_{k-1} along z^{k-1} by d_k.
3	Translate L_{k-1} along x^{k-1} by a_k.
4	Rotate L_{k-1} about x^{k-1} by α_k.

From Fig. 2-15 we see that the effect of operation 1 is to make axis x^{k-1} parallel to axis x^k. Following operation 2, axis x^{k-1} is then aligned (collinear) with axis x^k. Next, from Fig. 2-16 we observe that operation 3 ensures that the origins of frames L_{k-1} and L_k coincide. Finally, the effect of operation 4 is to align axis z^{k-1} with axis z^k. Thus the two coordinate frames L_{k-1} and L_k are coincident at this point. The four fundamental operations summarized in Table 2-3 each can be interpreted as a rotation or a translation of L_{k-1} along one of its own unit vectors. It follows from Algorithm 2-5-1 that we can postmultiply I by the four fundamental homogeneous transformation matrices to get the composite homogeneous transformation matrix associated with Table 2-3. We see from Table 2-3 that operations 1 and 2 together form a screw transformation along axis z^{k-1}. In view of Exercise 2-4-3, operations 3 and 4 also form a screw transformation, in this case along axis x^{k-1}. Consequently, the composite homogeneous transformation summarized in Table 2-3 can be ex-

pressed as a composition of two screw transformations, as follows:

$$T^{k}_{k-1}(\theta_k, d_k, a_k, \alpha_k) = \text{Screw } (d_k, \theta_k, 3) \text{ Screw } (a_k, \alpha_k, 1) \qquad (2\text{-}6\text{-}1)$$

Here T^{k}_{k-1} denotes the transformation from coordinate frame k to coordinate frame $k-1$. In general, T denotes a homogeneous coordinate transformation, the *superscript* being the index of the *source* coordinate frame and the *subscript* being the index of the *destination,* or reference, coordinate frame. As a consequence of the systematic notation for assigning link coordinates in the D-H algorithm, a single general expression for transforming between adjacent coordinate frames can be obtained. In particular, using Eq. (2-6-1) and the expression for the screw transformation in Def. 2-4-2, we arrive at the following result, which employs the shorthand notation, $Sx \triangleq \sin x$ and $Cx \triangleq \cos x$:

Proposition 2-6-1: Link-Coordinate Transformation. Let $\{L_0, L_1, \ldots, L_n\}$ be a set of link-coordinate frames assigned by Algorithm 2-5-1, and let $[q]^k$ and $[q]^{k-1}$ be the homogeneous coordinates of a point q with respect to frames L_k and L_{k-1}, respectively. Then, for $1 \leq k \leq n$, we have $[q]^{k-1} = T^{k}_{k-1}[q]^k$, where:

$$T^{k}_{k-1} = \begin{bmatrix} C\theta_k & -C\alpha_k S\theta_k & S\alpha_k S\theta_k & a_k C\theta_k \\ S\theta_k & C\alpha_k C\theta_k & -S\alpha_k C\theta_k & a_k S\theta_k \\ 0 & S\alpha_k & C\alpha_k & d_k \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Particular cases of the transformation matrix of Prop. 2-6-1 include T^{1}_{0}, which maps shoulder coordinates into base coordinates; T^{2}_{1}, which maps elbow coordinates into shoulder coordinates; T^{3}_{2}, which maps wrist or tool yaw coordinates into elbow coordinates, and so forth. If instead we want to move from the base toward the tool, we need to use the transformation T^{k-1}_{k}, which is the inverse of T^{k}_{k-1}.

Exercise 2-6-1: Inverse Link-Coordinate Transformation. Utilize Prop. 2-4-1 and Prop. 2-6-1 to show that the following transformation maps link $k-1$ coordinates into link k coordinates:

$$T^{k-1}_{k} = \begin{bmatrix} C\theta_k & S\theta_k & 0 & -a_k \\ -C\alpha_k S\theta_k & C\alpha_k C\theta_k & S\alpha_k & -d_k S\alpha_k \\ S\alpha_k S\theta_k & -S\alpha_k C\theta_k & C\alpha_k & -d_k C\alpha_k \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Recall that three of the kinematic parameters that appear in the coordinate transformation matrix T^{k}_{k-1} are constant, while the remaining parameter is the joint variable. The joint variable will sometimes be θ_k (for revolute joints) and sometimes be d_k (for prismatic joints). Rather than treat the two cases separately, we introduce a *joint type parameter,* defined as follows:

$$\xi_k \triangleq \begin{cases} 1 & \text{joint } k \text{ revolute} \\ 0 & \text{joint } k \text{ prismatic} \end{cases} \qquad (2\text{-}6\text{-}2)$$

Thus ξ_k is a binary-valued function of the joint type. With the aid of ξ_k we can then

define the kth *joint variable* q_k as follows:

$$q_k \triangleq \xi_k \theta_k + (1 - \xi_k) d_k \tag{2-6-3}$$

It is clear that ξ_k has the effect of selecting either θ_k or d_k, as appropriate. In general, the kth homogeneous link-coordinate transformation matrix T_{k-1}^k is a function of q_k for $1 \leq k \leq n$. For articulated-coordinate robots, we have $q = \theta$, but for all other robots q is a mixture of joint angles and joint distances. To solve the direct kinematics problem, we must determine the position and orientation of the tool relative to a coordinate frame attached to the base. The transformation from tool coordinates to base coordinates is obtained by a succession of coordinate transformations starting at the tool tip and working backward, one frame at a time, to the base. In particular, if $T_{\text{base}}^{\text{tool}}$ represents a transformation from tool-tip coordinates (link n) to base coordinates (link 0), then:

$$T_{\text{base}}^{\text{tool}}(q) = T_0^1(q_1) T_1^2(q_2) \cdots T_{n-1}^n(q_n) = T_0^n(q) \tag{2-6-4}$$

Note from the notation in Eq. (2-6-4) that when a sequence of coordinate transformations is performed, the coordinate transformation *algebra* is such that *diagonal indices cancel.* Consequently, for a composite transformation, the *superscript* on the rightmost factor is the identifier of the *source* coordinate frame, while the *subscript* on the leftmost factor is the identifier of the *destination,* or reference, coordinate frame.

In order to compute the arm matrix, it is often helpful to *partition* the problem at the wrist. This effectively decomposes the problem into two smaller subproblems, one subproblem associated with the major axes used to position the tool, and the other subproblem associated with the minor axes used to orient the tool. Here we break the expression for the arm matrix into two factors at the wrist, or third axis, as follows:

$$T_{\text{base}}^{\text{wrist}}(q) = T_0^1(q_1) T_1^2(q_2) T_2^3(q_3) = T_0^3(q) \tag{2-6-5}$$

$$T_{\text{wrist}}^{\text{tool}}(q) = T_3^4(q_4) T_4^5(q_5) \cdots T_{n-1}^n(q_n) = T_3^n(q) \tag{2-6-6}$$

For a general six-axis robot, $T_5^6(q_6)$ maps tool tip coordinates into roll coordinates, $T_4^5(q_5)$ maps roll coordinates into pitch coordinates, and $T_3^4(q_4)$ maps pitch coordinates into yaw coordinates or wrist coordinates. Thus the composite transformation $T_3^6(q)$ maps tool tip coordinates into wrist coordinates. Similarly, $T_2^3(q_3)$ maps wrist coordinates into elbow coordinates, $T_1^2(q_2)$ maps elbow coordinates into shoulder coordinates, and $T_0^1(q_1)$ maps shoulder coordinates into base coordinates. Thus the composite transformation $T_0^3(q)$ maps wrist coordinates into base coordinates. Once the two wrist-partitioned factors are obtained, the general arm matrix is then computed by forming their product, as follows:

$$T_{\text{base}}^{\text{tool}}(q) = T_{\text{base}}^{\text{wrist}}(q_1, q_2, q_3) T_{\text{wrist}}^{\text{tool}}(q_4, q_5, \ldots, q_n) \tag{2-6-7}$$

Again, notice that the diagonal identifiers in the composite transformation cancel. If one interchanges the subscript and superscript of a given transformation, this changes the *direction* of the transformation. Consequently, interchanging the subscript and the superscript is equivalent to *inverting* the transformation matrix, as in the following example:

$$T_{\text{tool}}^{\text{base}}(q) = [T_{\text{base}}^{\text{tool}}(q)]^{-1} \tag{2-6-8}$$

Recall from Prop. 2-4-1 that taking the inverse of a 4×4 homogeneous transformation matrix is a simple matter which involves transposing a 3×3 matrix and multiplying a 3×3 matrix by a 3×1 vector.

2-6-2 The Arm Equation

The reason for factoring the arm matrix into the product of two matrices partitioned at the wrist is that closed-form expressions for the two factors can be obtained relatively easily. At this point, the two matrices can be multiplied together and a closed-form expression for the entire arm matrix can then be obtained. Once an expression for the arm matrix is available, we can then substitute it into the following matrix equation, called the *arm equation:*

$$T_{\text{base}}^{\text{tool}}(q) = \left[\begin{array}{ccc:c} & R(q) & & p(q) \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{2-6-9}$$

For each value of the joint vector q, the arm matrix $T_{\text{base}}^{\text{tool}}(q)$ can be evaluated. The 3×3 upper left submatrix $R(q)$ specifies the *orientation* of the tool, while the 3×1 upper right submatrix $p(q)$ specifies the *position* of the tool tip. The three columns of R indicate the directions of the three unit vectors of the tool frame with respect to the base frame. Similarly, p specifies the coordinates of the tool tip with respect to the base frame. The solution of the direct kinematics problem in Eq. (2-6-9) is shown schematically in Fig. 2-21.

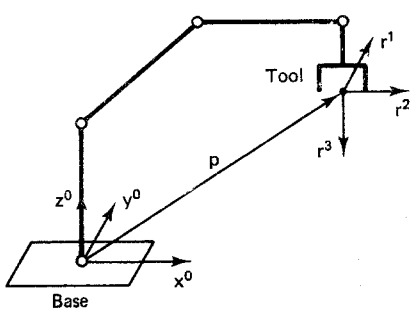


Figure 2-21 Position and orientation of the tool in base coordinates.

2-6-3 An Example: Microbot Alpha II

As an example of an arm matrix, consider the Alpha II robotic arm, whose link-coordinate diagram was developed in Fig. 2-19. First we find closed-form expressions for the two wrist-partitioned factors of the arm matrix. We begin by computing the wrist coordinates relative to the base. Using the notation $C_k \triangleq \cos q_k$ and $S_k \triangleq \sin q_k$ and some trigonometric identities from Appendix 1, we have from Prop. 2-6-1 and Table 2-2:

$$T^{\text{wrist}}_{\text{base}} = T^1_0 T^2_1 T^3_2 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_3 & -S_3 & 0 & a_3C_3 \\ S_3 & C_3 & 0 & a_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C_1C_2 & -C_1S_2 & -S_1 & a_2C_1C_2 \\ S_1C_2 & -S_1S_2 & C_1 & a_2S_1C_2 \\ -S_2 & -C_2 & 0 & d_1 - a_2S_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_3 & -S_3 & 0 & a_3C_3 \\ S_3 & C_3 & 0 & a_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C_1C_{23} & -C_1S_{23} & -S_1 & C_1(a_2C_2 + a_3C_{23}) \\ S_1C_{23} & -S_1S_{23} & C_1 & S_1(a_2C_2 + a_3C_{23}) \\ -S_{23} & -C_{23} & 0 & d_1 - a_2S_2 - a_3S_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-6-10}$$

Here, to simplify the notation, we have used $C_{kj} \triangleq \cos(q_k + q_j)$ and $S_{kj} \triangleq \sin(q_k + q_j)$.

Next, to compute the tool coordinates relative to the wrist, we have:

$$T^{\text{tool}}_{\text{wrist}} = T^4_3 T^5_4$$

$$= \begin{bmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C_4C_5 & -C_4S_5 & -S_4 & -d_5S_4 \\ S_4C_5 & -S_4S_5 & C_4 & d_5C_4 \\ -S_5 & -C_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-6-11}$$

Note that the final expression for $T^{\text{tool}}_{\text{wrist}}$ depends only on the last two joint variables. Similarly, the final expression for $T^{\text{wrist}}_{\text{base}}$ depends only on the first three joint variables. To find the arm matrix, we multiply the two wrist-partitioned factors together. The product, after simplification using the trigonometric identities in Appendix 1, and after substitution using the numerical values for d and a in Table 2-2, is the following arm matrix:

$$\left[\begin{array}{ccc:c} C_1C_{234}C_5 + S_1S_5 & -C_1C_{234}S_5 + S_1C_5 & -C_1S_{234} & C_1(117.8C_2 + 177.8C_{23} - 96.5S_{234}) \\ S_1C_{234}C_5 - C_1S_5 & -S_1C_{234}S_5 - C_1C_5 & -S_1S_{234} & S_1(177.8C_2 + 117.8C_{23} - 96.5S_{234}) \\ -S_{234}C_5 & S_{234}S_5 & -C_{234} & 215.9 - 177.8S_2 - 177.8S_{23} - 96.5C_{234} \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{2-6-12}$$

Here the notation C_{ijk} is short for $\cos(q_i + q_j + q_k)$ and the notation S_{ijk} is short for $\sin(q_i + q_j + q_k)$. Note that the approach vector r^3 and position vector p are independent of the tool roll angle q_5, as expected.

A FIVE-AXIS ARTICULATED ROBOT (RHINO XR-3)

In this section we perform a detailed kinematic analysis of a five-axis articulated robot with tool pitch and roll motions. This class of robots includes both the Rhino XR-3 educational robot and the Alpha II industrial robot as special cases. The Rhino XR-3 robot is shown in Fig. 2-22.

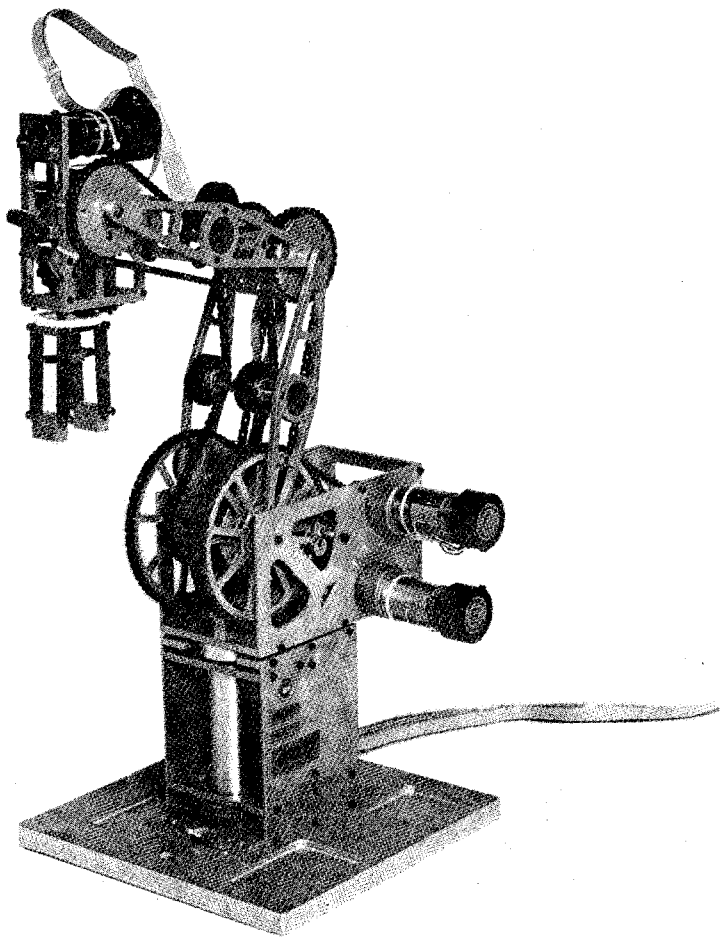


Figure 2-22 Five-axis articulated robot (Rhino XR-3). (Courtesy of Rhino Robots, Inc., Champaign, IL. Robotic arm developed and manufactured in the U.S.A.)

The physical structure of the Rhino XR-3 is such that the base motor is mounted vertically and remains fixed, while the shoulder, elbow, and tool pitch motors are mounted horizontally on the body, which rotates about the base axis. The tool roll and tool closure motors are smaller motors mounted in the hand.

2-7-1 The Link-Coordinate Diagram

We begin by constructing a link-coordinate diagram that is based on the Denavit-Hartenberg algorithm. Applying steps 0 to 7 of the D-H algorithm to the Rhino XR-3 robot, we get the diagram of link coordinates shown in Fig. 2-23.

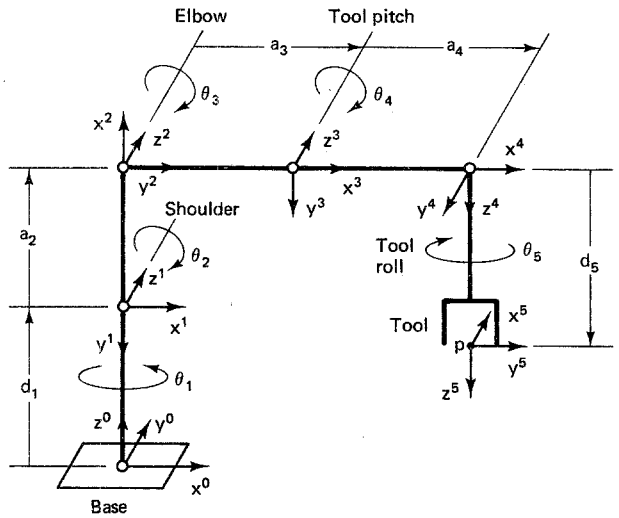


Figure 2-23 Link coordinates of a five-axis articulated robot (Rhino XR-3).

Next we apply steps 8 to 13 of the D-H algorithm starting with $k = 1$. Using Fig. 2-23, this yields the set of kinematic parameters shown in Table 2-4. Since this is an articulated-coordinate robot, the vector of joint variables is $q = \theta$. The values of q listed in the last column of Table 2-4 correspond to the *soft home* position pictured in the link-coordinate diagram of Fig. 2-23.

Note that d_5 represents the *tool length*, which may vary from robot to robot depending upon which type of tool is installed. The values for the joint distances d and link lengths a of the Rhino XR-3 robot are as follows (Hendrickson and Sandhu, 1986):

$$d = [26.04, 0.0, 0.0, 0.0, 16.83]^T \text{ cm} \tag{2-7-1}$$

$$a = [0.0, 22.86, 22.86, 0.95, 0.0]^T \text{ cm} \tag{2-7-2}$$

Here $d_5 = 16.83$ cm corresponds to the standard fingers. Optional fingers of different lengths are available. If the robot is mounted on the optional aluminum base, then $d_1 = 27.94$ cm.

2-7-2 The Arm Matrix

Next consider the problem of computing the arm matrix of the robot in Fig. 2-23. First we partition the problem at the wrist to generate two simpler subproblems. Using the notation $C_k \triangleq \cos q_k$ and $S_k \triangleq \sin q_k$, we have from Prop. 2-6-1 and Table 2-4:

$$
\begin{aligned}
T_{\text{base}}^{\text{wrist}} &= T_0^1 T_1^2 T_2^3 \\
&= \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & -S_2 & 0 & a_2 C_2 \\ S_2 & C_2 & 0 & a_2 S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_3 & -S_3 & 0 & a_3 C_3 \\ S_3 & C_3 & 0 & a_3 S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} C_1 C_2 & -C_1 S_2 & -S_1 & a_2 C_1 C_2 \\ S_1 C_2 & -S_1 S_2 & C_1 & a_2 S_1 C_2 \\ -S_2 & -C_2 & 0 & d_1 - a_2 S_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_3 & -S_3 & 0 & a_3 C_3 \\ S_3 & C_3 & 0 & a_3 S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} C_1 C_{23} & -C_1 S_{23} & -S_1 & C_1(a_2 C_2 + a_3 C_{23}) \\ S_1 C_{23} & -S_1 S_{23} & C_1 & S_1(a_2 C_2 + a_3 C_{23}) \\ -S_{23} & -C_{23} & 0 & d_1 - a_2 S_2 - a_3 S_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2-7-3}
$$

Here, to simplify the notation, we have used $C_{kj} = \cos(q_k + q_j)$ and $S_{kj} = \sin(q_k + q_j)$. The transformation $T_{\text{base}}^{\text{wrist}}$ provides the position and orientation of the wrist frame relative to the base frame. Since there is no tool yaw motion for this five-axis robot, the wrist frame corresponds to the tool pitch frame L_3. As a partial check of the expression for $T_{\text{base}}^{\text{wrist}}$, we can evaluate the transformation matrix at the soft home position. From Table 2-4, this is $q = [0, -\pi/2, \pi/2, 0, -\pi/2]^T$, which yields:

$$
T_{\text{base}}^{\text{wrist}}(\text{home}) = \left[\begin{array}{ccc:c} 1 & 0 & 0 & a_3 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d_1 + a_2 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{2-7-4}
$$

TABLE 2-4 KINEMATIC PARAMETERS OF A FIVE-AXIS ARTICULATED ROBOT

Axis	θ	d	a	α	Home
1	q_1	d_1	0	$-\pi/2$	0
2	q_2	0	a_2	0	$-\pi/2$
3	q_3	0	a_3	0	$\pi/2$
4	q_4	0	a_4	$-\pi/2$	0
5	q_5	d_5	0	0	$-\pi/2$

Inspection of Fig. 2-23 reveals that this is consistent with the link-coordinate diagram. The first column of the rotation matrix indicates that, in the soft home position, axis x^3 of the tool pitch frame has coordinates (1, 0, 0) in the base frame. Consequently, it is pointing in the same direction as base axis x^0. The second column of R indicates that axis y^3 of the tool pitch frame has coordinates (0, 0, −1) in the base frame, in which case it is pointing in the opposite direction of base axis z^0. The last

column of R specifies that axis z^3 of the tool pitch frame has coordinates (0, 1, 0) relative to the base. Thus z^3 points in the same direction as base axis y^0. Finally, the position vector p indicates that the coordinates of the origin of the tool pitch frame L_3 relative to the base frame L_0 are $(a_3, 0, d_1 + a_2)$. Thus L_3 is located a distance a_3 in front of the base and $d_1 + a_2$ above the base. This is consistent with Fig. 2-23.

Next we determine the position and orientation of the tool relative to the wrist:

$$
\begin{aligned}
T^{\text{tool}}_{\text{wrist}} &= T^4_3 T^5_4 \\
&= \begin{bmatrix} C_4 & 0 & -S_4 & a_4 C_4 \\ S_4 & 0 & C_4 & a_4 S_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} C_4 C_5 & -C_4 S_5 & -S_4 & a_4 C_4 - d_5 S_4 \\ S_4 C_5 & -S_4 S_5 & C_4 & a_4 S_4 + d_5 C_4 \\ -S_5 & -C_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{2-7-5}
$$

Note that the final expression for the position and orientation of the tool relative to the wrist depends only on the last two joint variables, as expected. At this point we could again make a partial check by evaluating this transformation at the home position. This should provide the position and orientation of frame L_5 relative to frame L_3. It remains to find the position and orientation of the tool relative to the base. This is obtained by multiplying the two wrist-partitioned factors together. After simplification using the trigonometric identities in Appendix 1, we get the following final result, which is the solution of the direct kinematics problem for the five-axis articulated robot in Fig. 2-23:

Proposition 2-7-1: Rhino XR-3. The arm matrix $T^{\text{tool}}_{\text{base}}(q)$ for the five-axis articulated-coordinate robot whose link-coordinate diagram is given in Fig. 2-23 is:

$$
\left[\begin{array}{ccc:c}
C_1 C_{234} C_5 + S_1 S_5 & -C_1 C_{234} S_5 + S_1 C_5 & -C_1 S_{234} & C_1(a_2 C_2 + a_3 C_{23} + a_4 C_{234} - d_5 S_{234}) \\
S_1 C_{234} C_5 - C_1 S_5 & -S_1 C_{234} S_5 - C_1 C_5 & -S_1 S_{234} & S_1(a_2 C_2 + a_3 C_{23} + a_4 C_{234} - d_5 S_{234}) \\
-S_{234} C_5 & S_{234} S_5 & -C_{234} & d_1 - a_2 S_2 - a_3 S_{23} - a_4 S_{234} - d_5 C_{234} \\
\hdashline
0 & 0 & 0 & 1
\end{array}\right]
$$

Note how the final expression for the position and orientation of the tool relative to the base now depends on all of the kinematic parameters. We can make a partial check by evaluating the arm matrix at the soft home position, which from Table 2-4 is $q = [0, -\pi/2, \pi/2, 0, -\pi/2]^T$. This yields:

$$
T^{\text{tool}}_{\text{base}}(\text{home}) = \left[\begin{array}{ccc:c}
0 & 1 & 0 & a_3 + a_4 \\
1 & 0 & 0 & 0 \\
0 & 0 & -1 & d_1 + a_2 - d_5 \\
\hdashline
0 & 0 & 0 & 1
\end{array}\right]
$$

Inspection of Fig. 2-23 reveals that this is consistent with the link-coordinate diagram. The first column of the rotation matrix indicates that, in the soft home position, tool axis x^5 (the normal vector) points in the same direction as axis y^0 of the base. The second column of R indicates that tool axis y^5 (the sliding vector) points in the same direction as axis x^0 of the base. The last column of R specifies that tool axis z^5 (the approach vector) points in the opposite direction from axis z^0 of the base. Finally, the position vector p indicates that the origin of the tool coordinate frame (the tool tip) is located a distance $a_3 + a_4$ in front of the base and $d_1 + a_2 - d_5$ above the base when the robot is in the soft home position.

The solution to the direct kinematics problem in Prop. 2-7-1 is *generic,* in the sense that all of the nonzero joint distances and link lengths appear as explicit parameters. Consequently, other robotic arms having a similar kinematic configuration but different values for these kinematic parameters can also be analyzed using this general expression for the arm matrix. Of course, not all five-axis articulated robots will fit this pattern. The Alpha II industrial robot is one that does, as can be seen from the following example.

Example 2-7-1: Alpha II Robot

As an example of the use of the generic form of the five-axis articulated arm matrix, consider the Alpha II robot, which is an articulated-coordinate robot that is sometimes used in wafer-handling applications in the semiconductor manufacturing industry. From Table 2-2, the link twist angles are the same as those in Table 2-4. However, we have the following joint distances and link lengths for the Alpha II robotic arm:

$$d = [215.9,\ 0.0,\ 0.0,\ 0.0,\ 96.5]^T \quad \text{mm}$$

$$a = [0.0,\ 177.8,\ 177.8,\ 0.0,\ 0.0]^T \quad \text{mm}$$

It is clear from these joint distances and link lengths that the Alpha II robot, unlike the Rhino XR-3 robot, has a spherical wrist ($a_4 = d_4 = 0$). Thus the arm matrix for the Alpha II is somewhat simpler than that for the Rhino XR-3. In particular, if we evaluate the generic expression for the arm matrix in Prop. 2-7-1 using the joint distances and link lengths for the Alpha II, the result is:

$$\left[\begin{array}{ccc|c} C_1C_{234}C_5 + S_1S_5 & -C_1C_{234}S_5 + S_1C_5 & -C_1S_{234} & C_1(177.8C_2 + 177.8C_{23} - 96.5S_{234}) \\ S_1C_{234}C_5 - C_1S_5 & -S_1C_{234}S_5 - C_1C_5 & -S_1S_{234} & S_1(177.8C_2 + 177.8C_{23} - 96.5S_{234}) \\ -S_{234}C_5 & S_{234}S_5 & -C_{234} & 215.9 - 177.8S_2 - 177.8S_{23} - 96.5C_{234} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

This is precisely the same result as obtained in Sec. 2-6, where an independent analysis of the kinematics of the Alpha II robot was performed. Thus, in terms of its kinematic structure, the Rhino XR-3 robot includes the Alpha II robot as a special case. As a partial check on this expression for the arm matrix for the Alpha II robot, we can evaluate it at the soft home position, which, from Table 2-2, corresponds to $q = [0,\ 0,\ 0,\ -\pi/2,\ 0]^T$. For these values of the joint variables, we have the following transformation matrix. Inspection of Fig. 2-19 reveals that this is indeed consistent with the link-coordinate diagram.

$$T^{\text{tool}}_{\text{base}}(\text{home}) = \left[\begin{array}{ccc:c} 0 & 0 & 1 & 485.1 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 215.9 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right]$$

2-7-3 Joint Coupling

It is apparent from the generic expression for the arm matrix in Prop. 2-7-1 that the angles $\theta_{23} = \theta_2 + \theta_3$ and $\theta_{234} = \theta_2 + \theta_3 + \theta_4$ appear repeatedly. From Fig. 2-23 we see that the angle θ_{23} can be thought of as a *global elbow* angle. It is the elbow angle measured relative to the *work surface* or x^0y^0 plane. When $\theta_{23} = 0$, the forearm is horizontal and pointing straight out, whereas when $\theta_{23} = -\pi/2$, it is vertical and pointing straight up. Similarly, the angle θ_{234} can be thought of as a *global tool pitch* angle. It is the tool pitch angle measured relative to the work surface or x^0y^0 plane. When $\theta_{234} = 0$, the tool is vertical and pointing straight down, whereas when $\theta_{234} = -\pi/2$ it is horizontal and pointing straight out.

The mechanical linkages for some robots, for example, the Rhino XR-3, are designed in such a way as to *couple* the movement of the shoulder, elbow, and tool pitch joints. When the shoulder motor is activated to produce a rotation of $\Delta\theta_2 = \beta$, the elbow moves simultaneously to produce a counteracting rotation of $\Delta\theta_3 = -\beta$. In this way the global elbow angle θ_{23} remains fixed when the shoulder motor is activated. Consequently, the orientation of the forearm relative to the base coordinate frame is unaffected by the shoulder motor. Similarly, when the elbow motor is activated to produce a rotation of $\Delta\theta_3 = \gamma$, the tool pitch moves simultaneously to produce a counteracting rotation of $\Delta\theta_4 = -\gamma$. Thus the global tool pitch angle θ_{234} remains fixed when either the shoulder motor or the elbow motor is activated. This means that the orientation of the tool relative to the base coordinate frame is unaffected by the shoulder and the elbow motors. These coupling effects can be useful when controlling the robot at the joint level. For example, if a liquid-filled container is carried by the tool, it will not spill on account of tipping when either the shoulder motor or the elbow motor is activated.

The kinematic analysis of a robotic arm using the D-H algorithm is based upon *independent* control of the joints with no coupling assumed between them. When mechanical coupling does exist, it can be removed at the software level by preprocessing the joint movement command $\Delta\theta$ with a coupling matrix C and a precision matrix P. This produces a command vector Δh that specifies the number of encoder counts to be sent to each motor starting with the base and ending with the tool roll. For the Rhino XR-3 robot, the encoder counts required to produce a movement of $\Delta\theta$ degrees in joint space can be formulated as follows:

$$\Delta h = \begin{bmatrix} \rho_1 & 0 & 0 & 0 & 0 \\ 0 & \rho_2 & 0 & 0 & 0 \\ 0 & 0 & -\rho_3 & 0 & 0 \\ 0 & 0 & 0 & -\rho_4 & 0 \\ 0 & 0 & 0 & 0 & \rho_5 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \Delta\theta = P^{-1}C\,\Delta\theta \qquad (2\text{-}7\text{-}7)$$

Here $\Delta\theta$ is a vector with five components which specifies the desired change in the five joint angles expressed in degrees. The coefficient matrix C is a *coupling* matrix. Note the 1s below the diagonal for the elbow command Δh_3 and the tool pitch command Δh_4. These off-diagonal terms effectively remove the coupling introduced by the parallel bar mechanism. The matrix P^{-1} is the inverse of a diagonal *precision* matrix. Here the ρ's specify the precision of each joint expressed in degrees per encoder count. From Table 1-11, we have:

$$\rho = [0.2269, 0.1135, 0.1135, 0.1135, 0.3125]^T \text{ degrees per count} \qquad (2\text{-}7\text{-}8)$$

Note that the terms ρ_3 and ρ_4 appearing in Eq. (2-7-7) have negative signs preceding them. This is because positive directions for the θ's in the link-coordinate diagram in Fig. 2-23 do not necessarily correspond to positive encoder counts in the Rhino movement command (Hendrickson and Sandhu, 1986). Given the way the Rhino XR-3 robots are wired at the factory, the base, shoulder, and tool roll motors have positive encoder counts corresponding to positive joint angles, but the elbow and tool pitch motors have negative encoder counts corresponding to positive joint angles; hence the minus signs to compensate. Of course, the Rhino XR-3 hardware could be modified by reversing the power leads to the motors and remounting (flipping) the optical encoder disks. However, this is really not necessary, since the introduction of minus signs in the software achieves the same result.

The vector Δh specifies the proper number of encoder hole counts to send to each of the five motors, starting with the base and ending with the tool roll, in order to achieve a move of $\Delta\theta$ degrees in joint space. A matrix formulation of Δh is used in Eq. (2-7-7) to highlight the interaxis coupling, which manifests itself through off-diagonal terms in C. For a software implementation of Eq. (2-7-7), clearly it would be more efficient to premultiply the matrix expression for Δh by hand to produce five separate scalar equations for the individual components of Δh.

2-8 A FOUR-AXIS SCARA ROBOT (ADEPT ONE)

Another important type of robotic manipulator is the four-axis horizontal-jointed robot, or SCARA robot. Examples of robots which belong to this general class include the Adept One robot, the IBM 7545 robot, the Intelledex 440 robot, and the Rhino SCARA robot. For illustration purposes we examine the Adept One robot, shown in Fig. 2-24. The treatment, however, is generic, and consequently the method used is applicable to SCARA robots in general.

The Adept One robot is unique in that it was the first commercial robot to implement a *direct drive* system for actuation. No gears or other mechanical power conversion devices are used. Instead, high-torque low-speed brushless DC motors are used to drive the joints directly. This eliminates gear friction and backlash and allows for clean, precise, high-speed operation.

2-8-1 The Link-Coordinate Diagram

To construct the link-coordinate diagram, we apply steps 0 to 7 of the D-H algorithm. The resulting set of link coordinates for the SCARA class of robots is shown in Fig. 2-25.

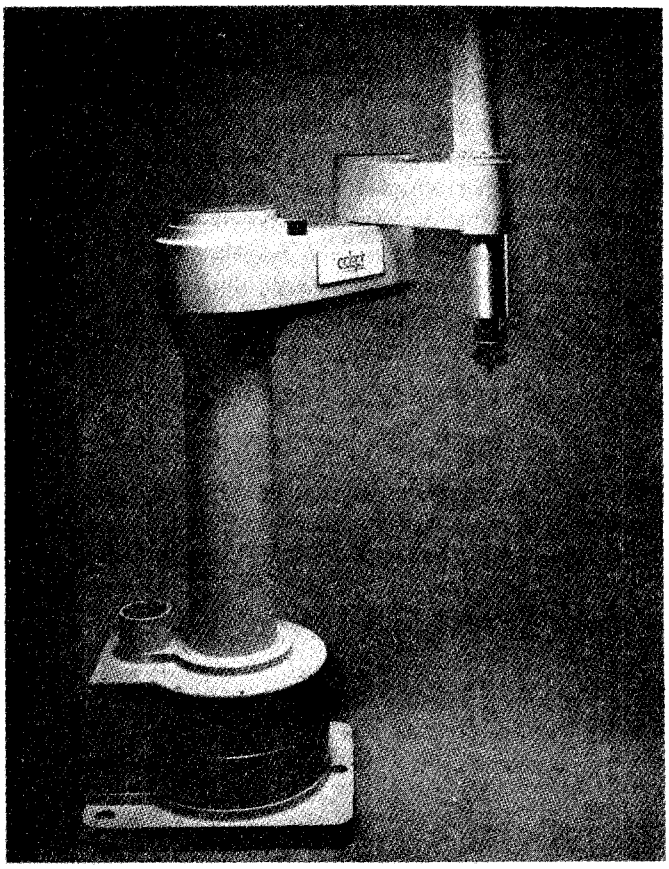


Figure 2-24 Four-axis SCARA robot (Adept One). (Courtesy of Adept Technology, Inc., San Jose, CA.)

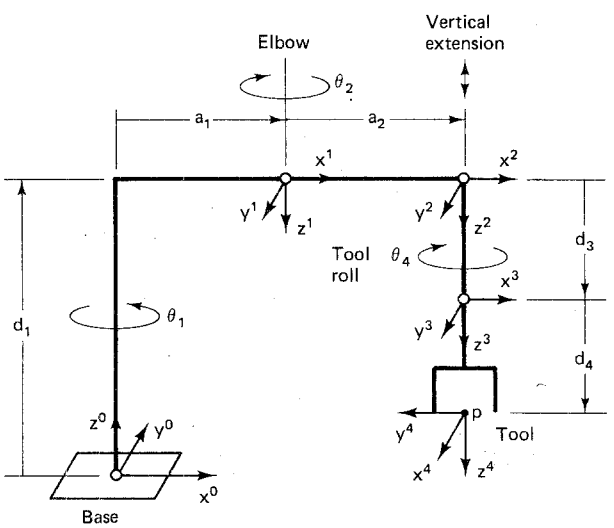


Figure 2-25 Link coordinates of a four-axis SCARA robot (Adept One)

In this case the vector of joint variables is $q = [\theta_1, \theta_2, d_3, \theta_4]^T$. The first two joint variables $\{\theta_1, \theta_2\}$ are revolute variables which establish the horizontal component of the tool position p. The third joint variable, d_3, is a prismatic variable which determines the vertical component of p. Finally, the last joint variable, θ_4, is a revolute variable which controls the tool orientation R. Applying steps 8 to 13 of the D-H algorithm yields the kinematic parameters shown in Table 2-5.

Note that 7 of the 12 constant parameters in Table 2-5 are zero, which makes the SCARA robot kinematically simple. Indeed, of all the practical industrial robots, the SCARA robot is perhaps the simplest to analyze. The values for the joint distances d and link lengths a of the Adept One robot are as follows:

$$d = [877, 0.0, d_3, 200]^T \quad \text{mm} \tag{2-8-1}$$

$$a = [425, 375, 0.0, 0.0]^T \quad \text{mm} \tag{2-8-2}$$

Here a tool length of $d_4 = 200$ mm has been assumed. Note that a specific value for d_3 has not been specified, because d_3 is a joint variable in this case. The values for d_3 range from 0 to 195 mm, which implies a vertical stroke of 195 mm.

2-8-2 The Arm Matrix

Next we compute the arm matrix $T^{\text{tool}}_{\text{base}}(q)$ for the SCARA robot. Since there are only four axes, we will not perform the intermediate step of partitioning the problem at the wrist. Instead we compute the arm matrix directly in one step, as follows:

$$T^{\text{tool}}_{\text{base}} = T^1_0 T^2_1 T^3_2 T^4_3$$

$$= \begin{bmatrix} C_1 & S_1 & 0 & a_1C_1 \\ S_1 & -C_1 & 0 & a_1S_1 \\ 0 & 0 & -1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_4 & -S_4 & 0 & 0 \\ S_4 & C_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C_{1-2} & S_{1-2} & 0 & a_1C_1 + a_2C_{1-2} \\ S_{1-2} & -C_{1-2} & 0 & a_1S_1 + a_2S_{1-2} \\ 0 & 0 & -1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_4 & -S_4 & 0 & 0 \\ S_4 & C_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C_{1-2} & S_{1-2} & 0 & a_1C_1 + a_2C_{1-2} \\ S_{1-2} & -C_{1-2} & 0 & a_1S_1 + a_2S_{1-2} \\ 0 & 0 & -1 & d_1 - q_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_4 & -S_4 & 0 & 0 \\ S_4 & C_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-8-3}$$

After multiplication and simplification using trigonometric identities from Appendix 1, we get the following arm matrix for a SCARA robot. Since the nonzero joint distances d and link lengths a have been left as explicit parameters, the follow-

TABLE 2-5 KINEMATIC PARAMETERS OF A FOUR-AXIS SCARA ROBOT

Axis	θ	d	a	α	Home
1	q_1	d_1	a_1	π	0
2	q_2	0	a_2	0	0
3	0	q_3	0	0	100
4	q_4	d_4	0	0	$\pi/2$

ing expression for the arm matrix is generic and applies to a variety of SCARA class robots:

Proposition 2-8-1: Adept One. The arm matrix $T_{\text{base}}^{\text{tool}}(q)$ for a four-axis SCARA robot whose link-coordinate diagram is given in Fig. 2-25 is:

$$T_{\text{base}}^{\text{tool}} = \left[\begin{array}{ccc:c} C_{1-2-4} & S_{1-2-4} & 0 & a_1C_1 + a_2C_{1-2} \\ S_{1-2-4} & -C_{1-2-4} & 0 & a_1S_1 + a_2S_{1-2} \\ 0 & 0 & -1 & d_1 - q_3 - d_4 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right]$$

Here the notation C_{1-2-4} denotes cos $(q_1 - q_2 - q_4)$, and similarly, S_{1-2-4} denotes sin $(q_1 - q_2 - q_4)$. Note that the approach vector is *fixed* at $r_3 = -i^3$, independent of the joint variables. This is characteristic of SCARA robots, which are designed to manipulate objects from directly above. They are often used in light assembly tasks such as the insertion of components into circuit boards.

Example 2-8-1: Adept One

Find the position of the tool tip of the Adept One robot when the joint variables are $q = [\pi/4, -\pi/3, 120, \pi/2]^T$.

Solution Using Prop. 2-8-1 and Eqs. (2-8-1) and (2-8-2), we have:

$$\begin{aligned} p &= [a_1C_1 + a_2C_{1-2}, a_1S_1 + a_2S_{1-2}, d_1 - q_3 - d_4]^T \\ &= [0.707a_1 - 0.259a_2, 0.707a_1 + 0.966a_2, d_1 - d_4 - 120]^T \\ &= [203.4, 662.7, 557.0]^T \text{ mm} \end{aligned}$$

2-9 A SIX-AXIS ARTICULATED ROBOT (INTELLEDEX 660)

As a third example, we perform a kinematic analysis of a six-axis articulated industrial robot, the Intelledex 660 manipulator, shown in Fig. 2-26. The Intelledex 660 robot is a high-precision light-assembly industrial robot that uses closed-loop stepper motor control to achieve a tool-tip repeatability of ±0.001 in. (±0.0254 mm). This type of robot finds use in a variety of assembly and material-handling applications, including wafer- and photomask-handling operations in clean rooms in the semiconductor manufacturing industry.

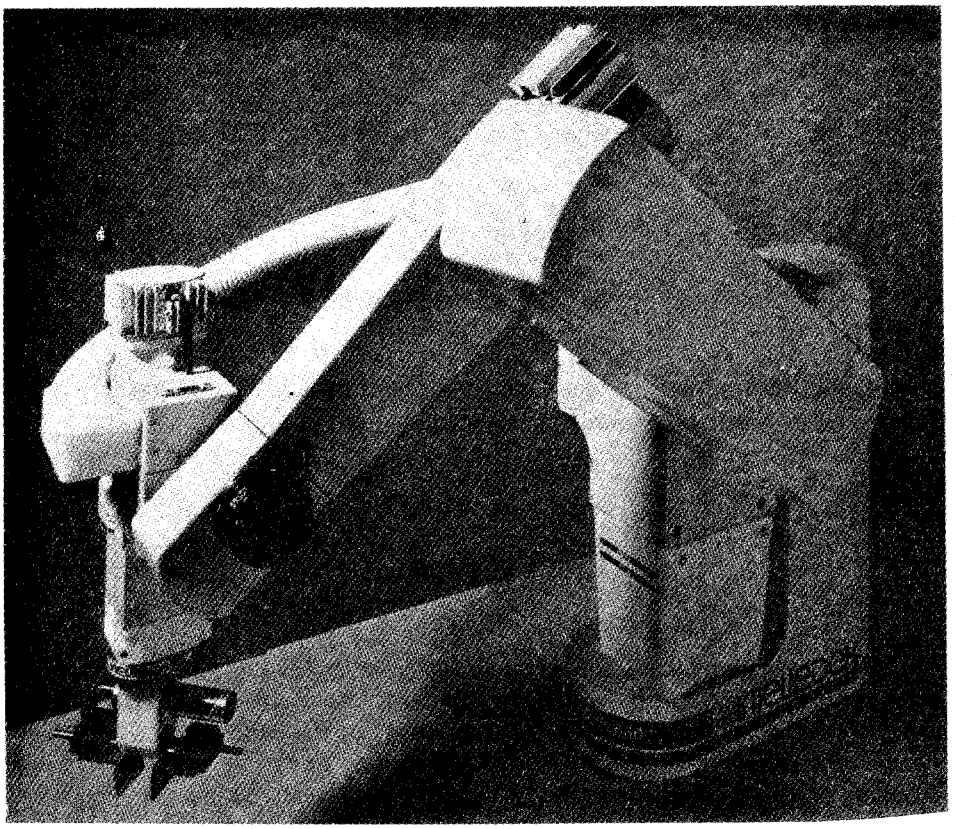


Figure 2-26 Six-axis articulated robot (Intelledex 660T). (Courtesy of Intelledex, Inc., Corvallis, OR.)

The physical structure of the Intelledex 660 robot is quite unique. It is a full six-axis robot, yet there are only two joints out near the end of the arm, a tool pitch joint ("outboard") and a tool roll joint ("tool spin"). Even though the wrist has only two joints, arbitrary orientations of the tool are possible. Tool yaw motion is achieved indirectly through a unique two-axis shoulder design. The first axis of the shoulder rolls the entire arm, while the second axis achieves a pitch-type motion in the plane selected by the shoulder roll. Thus, to implement a pure horizontal yaw motion of the tool, one can first roll the shoulder by $\pi/2$ radians and then activate the tool pitch joint. By effectively relocating the tool yaw joint in the shoulder, a reduction in the *weight* out at the end of the arm is realized. In some robots, such as the Microbot Alpha II and the GE P-50 robot, weight reduction at the end of the arm is achieved by placing the actuators for the distal joints back near the base of the robot and then running cables or chains up through the arm. The virtue of the Intelledex design is that there is no need for mechanical power transmission through the arm. Instead, the yaw motion is achieved indirectly through software.

2-9-1 The Link-Coordinate Diagram

First we construct a link-coordinate diagram based on the Denavit-Hartenberg algorithm. Applying steps 0 to 7 of the D-H algorithm to the Intelledex 660 robot, we get the diagram of link coordinates shown in Fig. 2-27.

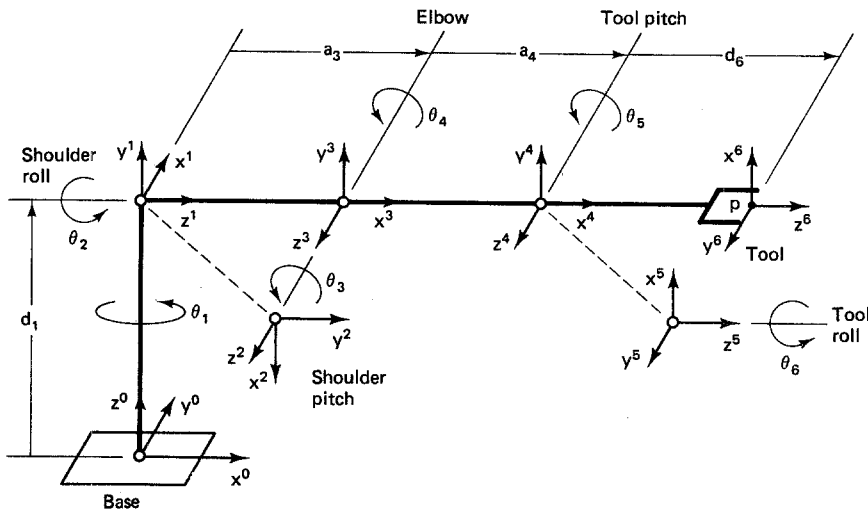


Figure 2-27 Link coordinates of a six-axis articulated robot (Intelledex 660T).

Next we apply steps 8 to 13 of the D-H algorithm starting with $k = 1$. Using Fig. 2-27, this yields the set of kinematic parameters shown in Table 2-6. Since this is an articulated-coordinate robot, the vector of joint variables is $q = \theta$. The values listed in Table 2-6 for θ correspond to the *soft home* position pictured in the link-coordinate diagram of Fig. 2-27.

TABLE 2-6 KINEMATIC PARAMETERS OF A SIX-AXIS ARTICULATED ROBOT

Axis	θ	d	a	α	Home
1	q_1	d_1	0	$\pi/2$	$\pi/2$
2	q_2	0	0	$\pi/2$	$-\pi/2$
3	q_3	0	a_3	0	$\pi/2$
4	q_4	0	a_4	0	0
5	q_5	0	0	$\pi/2$	$\pi/2$
6	q_6	d_6	0	0	0

Here 11 of the 18 fixed parameters are zero. In this case d_6 represents the *tool length,* which may vary from robot to robot, depending upon which type of tool is installed. The values for the joint distances d and link lengths a of the Intelledex 660 robot are as follows:

$$d = [373.4,\ 0.0,\ 0.0,\ 0.0,\ 0.0,\ 228.6]^T \quad \text{mm} \tag{2-9-1}$$

$$a = [0.0,\ 0.0,\ 304.8,\ 304.8,\ 0.0,\ 0.0]^T \quad \text{mm} \tag{2-9-2}$$

Here $d_6 = 228.6$ mm corresponds to the standard pneumatic gripper. If, instead, the small servo tool or some other tool is installed, then the calibration technique discussed in the operator's manual should be used to measure d_6.

2-9-2 The Arm Matrix

Next consider the problem of computing the arm matrix of the robot in Fig. 2-27. First we partition the problem after the third axis, which, in this case, is the elbow joint. Using Prop. 2-6-1 and Table 2-6, we have:

$$T_{\text{base}}^{\text{elbow}} = T_0^1 T_1^2 T_2^3$$

$$= \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & 0 & S_2 & 0 \\ S_2 & 0 & -C_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_3 & -S_3 & 0 & a_3 C_3 \\ S_3 & C_3 & 0 & a_3 S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C_1 C_2 & S_1 & C_1 S_2 & 0 \\ S_1 C_2 & -C_1 & S_1 S_2 & 0 \\ S_2 & 0 & -C_2 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_3 & -S_3 & 0 & a_3 C_3 \\ S_3 & C_3 & 0 & a_3 S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C_1 C_2 C_3 + S_1 S_3 & -C_1 C_2 S_3 + S_1 C_3 & C_1 S_2 & (C_1 C_2 C_3 + S_1 S_3)a_3 \\ S_1 C_2 C_3 - C_1 S_3 & -S_1 C_2 S_3 - C_1 C_3 & S_1 S_2 & (S_1 C_2 C_3 - C_1 S_3)a_3 \\ S_2 C_3 & -S_2 S_3 & -C_2 & d_1 + S_2 C_3 a_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2-9-3}$$

The transformation $T_{\text{base}}^{\text{elbow}}$ provides the position and orientation of the elbow frame L_3 relative to the base frame L_0. As a partial check of the expression for $T_{\text{base}}^{\text{elbow}}$, we can evaluate it at the soft home position. From the last column of Table 2-6, this is $q = [\pi/2,\ -\pi/2,\ \pi/2,\ 0,\ \pi/2,\ 0]^T$, which yields:

$$T_{\text{base}}^{\text{elbow}}(\text{home}) = \left[\begin{array}{ccc:c} 1 & 0 & 0 & a_3 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & d_1 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{2-9-4}$$

Inspection of Fig. 2-27 reveals that this specification of the position and orientation of frame L_3 relative to frame L_0 is consistent with the link-coordinate diagram.

Next we determine the position and orientation of the tool relative to the elbow:

$$
\begin{aligned}
T_{\text{elbow}}^{\text{tool}} &= T_3^4 T_4^5 T_5^6 \\
&= \begin{bmatrix} C_4 & -S_4 & 0 & a_4 C_4 \\ S_4 & C_4 & 0 & a_4 S_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} C_{45} & 0 & S_{45} & a_4 C_4 \\ S_{45} & 0 & -C_{45} & a_4 S_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} C_{45}C_6 & -C_{45}S_6 & S_{45} & a_4 C_4 + S_{45} d_6 \\ S_{45}C_6 & -S_{45}S_6 & -C_{45} & a_4 S_4 - C_{45} d_6 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2-9-5}
$$

Note that the final expression for the position and orientation of the tool relative to the elbow depends only on the last three joint variables, as expected. At this point we could again make a partial check by evaluating this transformation at the soft home position. This should provide the position and orientation of frame L_6 relative to frame L_3. We still need to find the position and orientation of the tool relative to the base. This is obtained by multiplying the two elbow-partitioned factors together. After considerable simplification using the trigonometric identities in Appendix 1, we get the following final result, which is the solution of the direct kinematics problem for the six-axis Intelledex 660 robot:

Proposition 2-9-1: Intelledex 660. The position $p(q)$ and orientation $R(q)$ of the tool frame relative to the base frame for the six-axis articulated-coordinate robot whose link-coordinate diagram is given in Fig. 2-27 are:

$$
p = \begin{bmatrix} C_1 C_2(a_3 C_3 + a_4 C_{34} + d_6 S_{345}) + S_1(a_3 S_3 + a_4 S_{34} - d_6 C_{345}) \\ S_1 C_2(a_3 C_3 + a_4 C_{34} + d_6 S_{345}) - C_1(a_3 S_3 + a_4 S_{34} - d_6 C_{345}) \\ d_1 + S_2(a_3 C_3 + a_4 C_{34} + d_6 S_{345}) \end{bmatrix}
$$

$$
R = \left[\begin{array}{c|c|c} (C_1 C_2 C_{345} + S_1 S_{345})C_6 + C_1 S_2 S_6 & C_1 S_2 C_6 - (C_1 C_2 C_{345} + S_1 S_{345})S_6 & -S_1 C_{345} + C_1 C_2 S_{345} \\ (S_1 C_2 C_{345} - C_1 S_{345})C_6 + S_1 S_2 S_6 & S_1 S_2 C_6 - (S_1 C_2 C_{345} - C_1 S_{345})S_6 & C_1 C_{345} + S_1 C_2 S_{345} \\ S_2 C_{345} C_6 - C_2 S_6 & -C_2 C_6 - S_2 C_{345} S_6 & S_2 S_{345} \end{array}\right]
$$

It is evident that the expressions for the approach vector $r^3(q)$ and the tool-tip position $p(q)$ are independent of the tool roll angle q_6, as expected. We can make a partial check of the final expression for the arm matrix by evaluating it at the soft home position, which corresponds to $q = [\pi/2, -\pi/2, \pi/2, 0, \pi/2, 0]^T$. This yields:

$$T_{\text{base}}^{\text{tool}}(\text{home}) = \left[\begin{array}{ccc:c} 0 & 0 & 1 & a_3 + a_4 + d_6 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & d_1 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{2-9-6}$$

Note that this is consistent with the link-coordinate diagram in Fig. 2-27. The first column of the rotation matrix indicates that, in the soft home position, tool axis x^6 (the normal vector) points in the same direction as axis z^0 of the base. The second column of R indicates that tool axis y^6 (the sliding vector) points in the opposite direction from axis y^0 of the base. The last column of R specifies that tool axis z^6 (the approach vector) points in the same direction as axis x^0 of the base. Finally, the position vector p indicates that the origin of the tool coordinate frame (the tool tip) is located a distance $a_3 + a_4 + d_6$ in front of the base and d_1 above the base when the robot is in the soft home position.

2-10 PROBLEMS

2-1. Which of the kinematic parameters are variable for a revolute joint? Which are variable for a prismatic joint?

2-2. Consider the single-axis robot in Fig. 2-28 shown in the home position, which corresponds to $\theta = \pi/2$. Suppose the point p on the mobile link has coordinates $[p]^M = [0.5, 0.5, 2.0]^T$.

(a) Find an expression for $R(\theta)$, the coordinate transformation matrix which maps mobile M coordinates into fixed F coordinates as a function of the joint variable θ.

(b) Use $R(\theta)$ to find $[p]^F$ when $\theta = \pi$ and when $\theta = 0$.

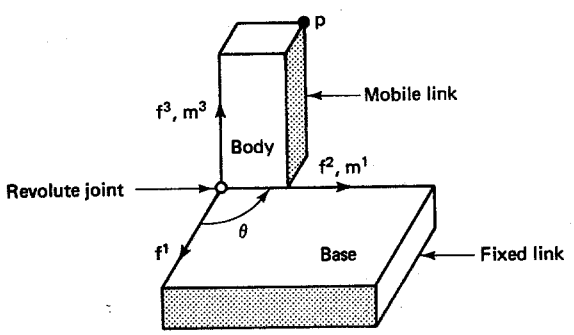


Figure 2-28 Single-axis robot.

2-3. Consider the following coordinate transformation matrix, which represents a fundamental rotation. What is the axis of rotation (1, 2, or 3), and what is the angle of rotation?

$$R = \begin{bmatrix} 0.500 & 0 & -0.866 \\ 0 & 1 & 0 \\ 0.866 & 0 & 0.500 \end{bmatrix}$$

2-4. Consider the robotic tool shown in Fig. 2-29. Suppose we yaw the tool by π about f^1, then pitch the tool by $-\pi/2$ about f^2, and finally roll the tool by $\pi/2$ about f^3.

(a) Sketch the sequence of tool positions after each of the yaw, pitch, and roll movements.

(b) Find the transformation matrix T which maps tool coordinates M into wrist coordinates F following the sequence of rotations.

(c) Find $[p]^F$, the location of the tool tip p in wrist coordinates F following the sequence of rotations, assuming that the tool-tip coordinates in terms of the tool frame are $[p]^M = [0, 0, 0.8]^T$.

2-5. Do Prob. 2-4 but with the yaw, pitch, and roll of the tool performed in reverse order about the unit vectors of the M coordinate frame rather than about the unit vectors of the F coordinate frame.

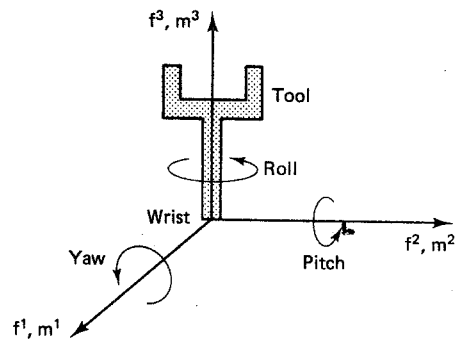


Figure 2-29 Yaw, pitch, and roll of tool.

2-6. Consider the following composite homogeneous coordinate transformation matrix T:

$$T = \left[\begin{array}{ccc:c} & R & & p \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right]$$

Assuming that R and p represent a rotation and a translation of the mobile frame about its own unit vectors, which of the following sequence of operations does the composite transformation T represent? That is, what is the *implicit* order of the two fundamental operations?

(a) T represents a rotation of R followed by a translation of p?

(b) T represents a translation of p followed by a rotation of R?

(c) T represents neither in general?

2-7. Find a numerical example which shows that the order in which a rotation and a translation are performed does affect the final relationship between two initially coincident coordinate frames. Specify the coordinate transformation matrices, and sketch the sequence of positions.

2-8. Homogeneous coordinate transformation matrices can be used to represent scaling operations as well as rotations and translations. Consider the following *fundamental homogeneous scaling matrix:*

$$\text{Scale } (c, \sigma) \triangleq \left[\begin{array}{ccc:c} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ 0 & 0 & c_3 & 0 \\ \hdashline 0 & 0 & 0 & \sigma \end{array}\right]$$

(a) Show that when $\sigma \neq 0$ and $c = [1, 1, 1]^T$, Scale (c, σ) scales all three physical coordinates by $1/\sigma$. Here σ is referred to as a *global* scale factor.

(b) Show that Scale $(c, 1)$ scales the kth physical coordinate by c_k for $1 \leq k \leq 3$. Here c is referred to as a vector of *local* scale factors.

(c) Show that when $\sigma \neq 0$ and $c = [\sigma, \sigma, \sigma]^T$, the local and global scale factors cancel one another, in which case multiplication by Scale (c, σ) has no effect on the physical coordinates.

2-9. Consider the Unimation PUMA 200 manipulator, shown in Fig. 2-30. This is a six-axis articulated robot with a roll-pitch-roll type of spherical wrist. Assign link coordinates using the first half of the D-H algorithm. Label the diagram with a's and d's as appropriate.

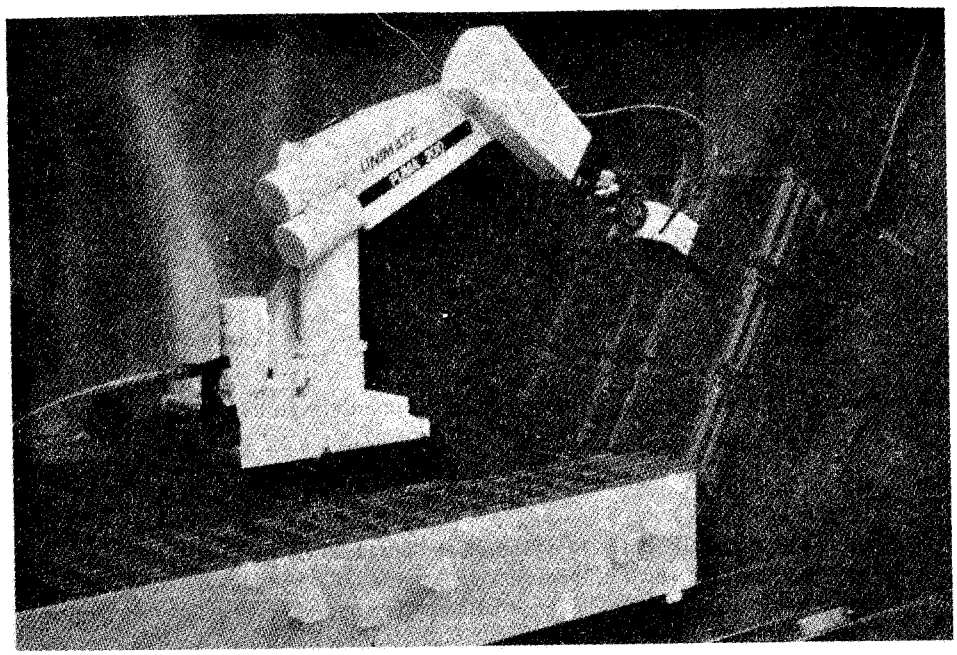


Figure 2-30 Unimation PUMA 200 robot. (Courtesy of Westinghouse Automation Division, Pittsburgh, PA.)

2-10. Use the second half of the D-H algorithm to fill in the kinematic parameters for the PUMA 200 robot in Table 2-7, consistent with your link-coordinate diagram from Prob. 2-9. Indicate which parameters are the joint variables.

TABLE 2-7 PUMA 200 KINEMATIC PARAMETERS

Axis	θ	d	a	α	Home
1					
2					
3					
4					
5					
6					

2-11. Compute the first factor of the arm matrix $T_{base}^{wrist}(q)$ for the PUMA 200 robot using your table of kinematic parameters from Prob. 2-10.

2-12. Compute the second factor of the arm matrix $T_{wrist}^{tool}(q)$ for the PUMA 200 robot using your table of kinematic parameters from Prob. 2-10.

2-13. Compute the arm matrix $T_{base}^{tool}(q)$ for the PUMA 200 robot using your wrist-partitioned factors from Prob. 2-11 and Prob. 2-12.

2-14. An alternative way to specify the tool orientation is the *Z-Y-Z* Euler angle representation. The mobile frame M and fixed frame F start out coincident. Frame M is first rotated about m^3 by an angle of α; it is then rotated about m^2 by an angle of β; and finally, frame M is again rotated about m^3 by an angle of γ. Find the composite *Z-Y-Z* Euler angle rotation matrix $R_{zyz}(\alpha, \beta, \gamma)$ which maps mobile M coordinates into fixed F coordinates.

2-15. Consider the United States Robots Maker 110 manipulator shown in Fig. 2-31. This is a five-axis spherical coordinate robot with a pitch-roll spherical wrist. Assign link coordinates using the first half of the D-H algorithm. Label the diagram with a's and d's as appropriate.

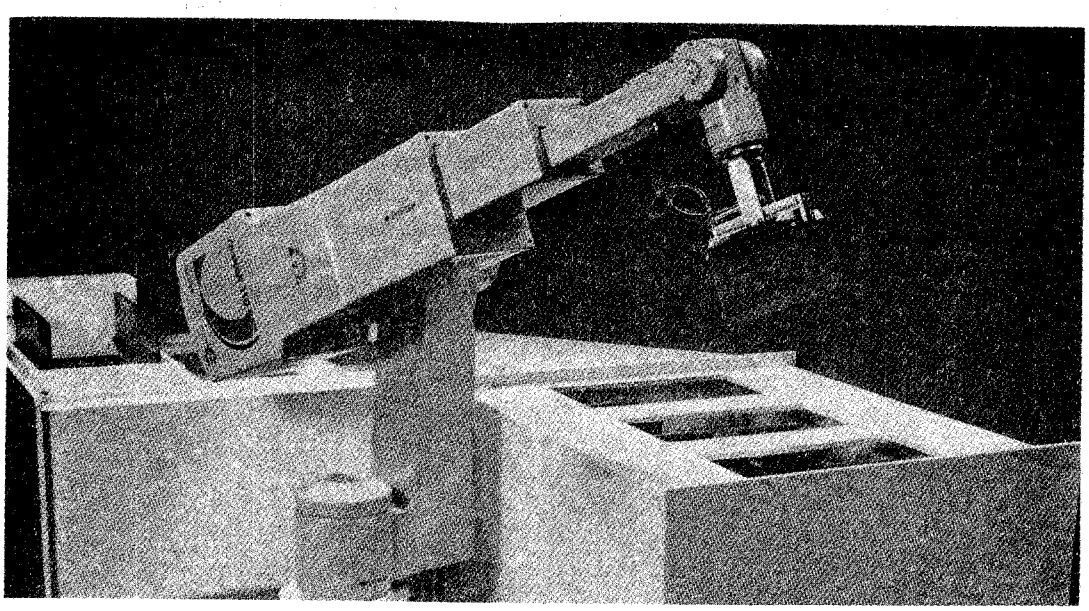


Figure 2-31 United States Robots Maker 110 robot. (Courtesy of United States Robots, King of Prussia, PA.)

2-16. Use the last half of the D-H algorithm to fill in the table of kinematic parameters for the Maker 110 in Table 2-8, consistent with your link-coordinate diagram for Prob. 2-15. Indicate which parameters are the joint variables.

2-17. Compute the first factor of the arm matrix $T_{base}^{wrist}(q)$ for the Maker 110 robot using your table of kinematic parameters from Prob. 2-16.

2-18. Compute the second factor of the arm matrix $T_{wrist}^{tool}(q)$ for the Maker 110 robot using your table of kinematic parameters from Prob. 2-16.

TABLE 2-8 MAKER 110 KINEMATIC PARAMETERS

Axis	θ	d	a	α	Home
1					
2					
3					
4					
5					

2-19. Compute the arm matrix $T_{\text{base}}^{\text{tool}}(q)$ for the Maker 110 robot using your wrist-partitioned factors from Prob. 2-17 and Prob. 2-18.

REFERENCES

CRAIG, J. (1986). *Introduction to Robotics: Mechanics and Control,* Addison-Wesley: Reading, Mass.

DENAVIT, J., and R. S. HARTENBERG (1955). "A kinematic notation for lower-pair mechanisms based on matrices," *ASME J. Appl. Mech.*, June, pp. 215–221.

FU, K. S., R. C. GONZALEZ, and C. S. G. LEE (1987). *Robotics: Control, Sensing, Vision, and Intelligence,* McGraw-Hill: New York.

HENDRICKSON, T., and H. SANDHU (1986). *XR-3 Robot Arm, Mark III 8 Axis Controller: Owner's Manual,* Version 3.00, Rhino Robots, Inc.: Champaign, Ill.

NOBLE, B. (1969). *Applied Linear Algebra,* Prentice-Hall, Inc.: Englewood Cliffs, N.J.

PAUL, R. P. (1982). *Robot Manipulators: Mathematics, Programming, and Control,* MIT Press: Cambridge, Mass.

SCHILLING, R. J. and H. LEE (1988). *Engineering Analysis: A Vector Space Approach,* Wiley: New York.

3

Inverse Kinematics: Solving The Arm Equation

In Chap. 2 we developed a procedure for determining the position and orientation of the tool of a robotic manipulator given the vector of joint variables. We now examine the inverse problem of determining the joint variables given a desired position and orientation for the tool. The inverse kinematics problem is important because manipulation tasks are naturally formulated in terms of the desired tool position and orientation. This is the case, for example, when external sensors such as overhead cameras are used to plan robot motion. The information provided by the camera is not in terms of joint variables; it specifies the positions and orientations of the objects that are to be manipulated.

The inverse kinematics problem is more difficult than the direct kinematics problem because a systematic closed-form solution applicable to robots in general is not available. Moreover, when closed-form solutions to the arm equation can be found, they are seldom unique. The ways in which multiple solutions arise are investigated. A compact representation of tool position and orientation called the tool-configuration vector is then introduced. Solutions to the arm equation for four generic classes of robots are presented. Representative members of these classes include: the five-axis Rhino XR-3, the four-axis Adept One, the six-axis Intelledex 660, and a three-axis planar articulated arm. Chapter 3 concludes with an example of an application of inverse kinematics to plan motion in a robotic work cell using information available from an overhead camera.

3-1 THE INVERSE KINEMATICS PROBLEM

The key to the solution of the direct kinematics problem was the Denavit-Hartenberg (D-H) algorithm, a systematic procedure for assigning link coordinates to a robotic manipulator. Successive transformations between adjacent coordinate frames, start-

ing at the tool tip and working back to the base of the robot, then led to the arm matrix. The arm matrix represents the position p and orientation R of the tool in the base frame as a function of the joint variables q, as shown in Fig. 3-1.

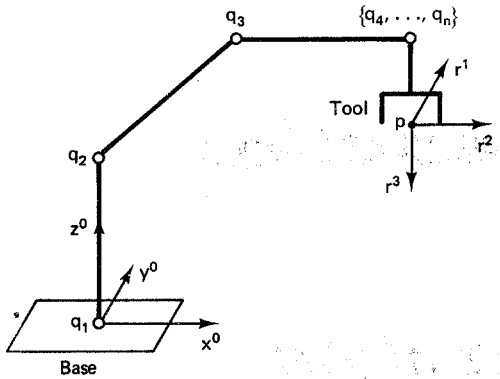


Figure 3-1 Tool configuration as a function of joint variables.

For convenience, we will refer to the position and orientation of the tool collectively as the *configuration* of the tool. The solution to the direct kinematics problem can be expressed in the following form, where R represents the *rotation* and p represents the *translation* of the tool frame relative to the base frame:

$$T_{\text{base}}^{\text{tool}}(q) = \left[\begin{array}{ccc|c} & R & & p \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3-1-1}$$

The solution to the direct kinematics problem is useful because it provides us with a relationship which explicitly shows the dependence of the tool configuration on the joint variables. This can be utilized, for example, in determining the size and shape of the work envelope. In particular, suppose Q represents the range of values in $\mathbf{R}^n$ that the joint variables can assume. We refer to Q as the *joint-space work envelope* of the robot. Typically, the joint-space work envelope is a convex polyhedron in $\mathbf{R}^n$ of the following general form:

$$Q \triangleq \{q \in \mathbf{R}^n: q^{\min} \leq Cq \leq q^{\max}\} \tag{3-1-2}$$

Here $q^{\min}$ and $q^{\max}$ are constant vectors in $\mathbf{R}^m$ which represent *joint limits* and C is a *joint coupling* matrix. For the simplest special case, the joint coupling matrix is $C = I$; in this case $q_k^{\min}$ and $q_k^{\max}$ represent lower and upper limits, respectively, for joint k. More generally, if the joint coupling matrix C has some off-diagonal terms, then Eq. (3-1-2) represents constraints on linear combinations of joint variables. In this case the number of inequality constraints m may exceed the number of variables n. The set of positions in Cartesian space $\mathbf{R}^3$ that are reachable by the tool tip can be investigated by examining the values of $p(q)$ as q ranges over Q. Similarly, the set of orientations achievable by the tool can be determined by examining the values of $R(q)$ as q ranges over the joint-space work envelope Q.

Perhaps the most important benefit provided by the solution of the direct kinematics problem is that it lays a foundation for solving a related problem, the inverse

kinematics problem. The vector of joint variables q is restricted to the subset Q of $\mathbf{R}^n$. We refer to the vector space $\mathbf{R}^n$ in this case as *joint space*. Similarly, the tool-configuration parameters $\{R, p\}$ can be associated with a subset W of $\mathbf{R}^6$. We refer to the vector space $\mathbf{R}^6$ in this case as *tool-configuration space*. Tool-configuration space is six-dimensional because arbitrary configurations of the tool can be specified by using three position coordinates (p_1, p_2, p_3) together with three orientation coordinates (yaw, pitch, roll). Solving the direct kinematics problem is equivalent to finding the mapping from joint space to tool-configuration space, while solving the inverse kinematics problem is equivalent to finding an inverse mapping from tool-configuration space back to joint space. A pictorial summary of the relationship between the two forms of the kinematics problem is shown in Fig. 3-2.

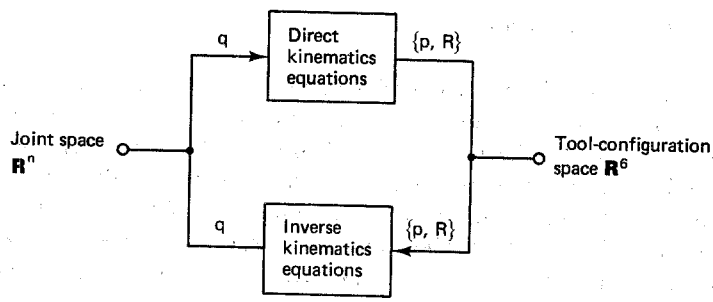


Figure 3-2 Direct and inverse kinematics.

The inverse kinematics problem is more difficult than the direct kinematics problem, because no single explicit systematic procedure analogous to the D-H algorithm is available. As a result, each robot or generically similar class of robots has to be treated separately. However, the solution to the inverse kinematics problem is much more useful. Indeed, a key to making robots more versatile lies in using feedback from external sensors such as tactile arrays and vision. External sensors supply information about part location and part orientation directly in terms of configuration-space variables. It is then necessary to use this information to determine appropriate values for the joint variables to properly configure the tool. Thus we must find a mapping from a tool-configuration space input specification into a joint-space output specification. This is the inverse kinematics problem. We characterize this problem more formally as follows:

Problem: Inverse Kinematics. Given a desired position p and orientation R for the tool, find values for the joint variables q which satisfy the arm equation in Eq. (3-1-1).

The solution to the inverse kinematics problem is useful even when external sensors are not employed. A case in point is the problem of getting the end-effector or tool to follow a straight-line path. Certain types of assembly, welding, and sealing operations require that straight-line paths be followed or at least closely approximated. A straight-line trajectory is naturally formulated in tool-configuration space. It is then necessary to find a corresponding trajectory in joint space which will pro-

duce the straight-line motion of the tool tip. This is an important special case of the inverse kinematics problem.

3-2 GENERAL PROPERTIES OF SOLUTIONS

Although a general solution has been obtained for the direct kinematics problem, the details of an explicit solution to the inverse kinematics problem depend upon the robot or the class of robots being investigated. However, there are certain characteristics of the solution that hold in general.

3-2-1 Existence of Solutions

First let us examine conditions under which solutions to the inverse kinematics problem exist. Clearly, if the desired tool-tip position p is outside its work envelope, then no solution can exist. Furthermore, even when p is within the work envelope, there may be certain tool orientations R which are not realizable without violating one or more of the joint variable limits. Indeed, if the robot has fewer than three degrees of freedom to orient the tool, then whole classes of orientations are unrealizable. To examine this issue further, consider the following more detailed formulation of the arm equation:

$$T_{\text{base}}^{\text{tool}}(q) = \left[\begin{array}{ccc:c} R_{11} & R_{12} & R_{13} & p_1 \\ R_{21} & R_{22} & R_{23} & p_2 \\ R_{31} & R_{32} & R_{33} & p_3 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{3-2-1}$$

Since the last row of the arm matrix is always constant, the arm equation constitutes a system of 12 simultaneous nonlinear algebraic equations in the n unknown components of q. However, the 12 equations are by no means independent of one another. Since R is a coordinate transformation matrix representing a pure rotation from one orthonormal frame to another, it follows from Prop. 2-2-3 that $R^{-1} = R^T$. But if $R^TR = I$, then the three columns of R form an orthonormal set. The mutual orthogonality puts three constraints on the columns of R, namely:

$$r^1 \cdot r^2 = 0 \tag{3-2-2}$$

$$r^1 \cdot r^3 = 0 \tag{3-2-3}$$

$$r^2 \cdot r^3 = 0 \tag{3-2-4}$$

These constraints come from the off-diagonal terms of $R^TR = I$. The diagonal terms dictate that each column of R must also be a unit vector. This adds three more constraints to the columns of R:

$$\|r^k\| = 1 \qquad 1 \leq k \leq 3 \tag{3-2-5}$$

Thus the 12 constraints implicit in the arm equation actually represent only six independent constraints on the n unknown components of the vector of joint variables q. If we are to have a general solution to the inverse kinematics problem, one for which

a q can be found which generates an arbitrary tool configuration, then the number of unknowns must at least match the number of independent constraints. That is:

$$\text{General manipulation} \Rightarrow n \geq 6 \tag{3-2-6}$$

This lower bound on the number of axes n is a necessary but not sufficient condition for the existence of a solution to the inverse kinematics problem when arbitrary tool configurations are specified. Clearly, the tool position must be within the work envelope of the robot, and the tool orientation must be such that none of the limits on the joint variables are violated. Even when these additional constraints on the values of p and R are satisfied, there is no guarantee that a *closed-form expression* for a solution to the inverse kinematics problem can be obtained (Pieper, 1968).

The general strategy for solving the inverse kinematics problem simplifies somewhat when the robot has a *spherical wrist*. To see this, consider the case of an n-axis robot where $4 \leq n \leq 6$. Suppose the last axis is a tool roll axis, and suppose the robot has a spherical wrist, which means that the $n - 3$ axes at the end of the arm all intersect at a point. For this class of robots, the inverse kinematics problem can be decomposed into two smaller subproblems by partitioning the original problem at the wrist. Given the tool tip position p and tool orientation R, the *wrist position* p^{wrist} can be inferred from p by working backward along the approach vector:

$$p^{\text{wrist}} = p - d_n r^3 \tag{3-2-7}$$

Here the joint distance d_n represents the *tool length* for an n-axis robot as long as the last axis is a tool roll axis. The approach vector r^3 is simply the third column of the rotation matrix R. Once the wrist position p^{wrist} is obtained from $\{p, R, d_n\}$, the first three joint variables $\{q_1, q_2, q_3\}$ that are used to position the wrist can then be obtained from the following *reduced arm equation:*

$$T^{\text{wrist}}_{\text{base}}(q_1, q_2, q_3) i^4 = \begin{bmatrix} p - d_n r^3 \\ 1 \end{bmatrix} \tag{3-2-8}$$

The fourth column of $T^{\text{wrist}}_{\text{base}}$ represents the homogeneous coordinates of the origin of the wrist frame L_3 relative to the base frame L_0. Since the wrist coordinates depend only on the joint variables $\{q_1, q_2, q_3\}$, these joint variables of the major axes can be solved for *separately* using Eq. (3-2-8). Once the major axis variables $\{q_1, q_2, q_3\}$ are found, their values can then be substituted into the general arm equation in Eq. (3-2-1) and it can be solved for the remaining tool orientation variables $\{q_4, \ldots, q_n\}$.

3-2-2 Uniqueness of Solutions

The existence of a solution to the inverse kinematics problem is not the only issue that needs to be addressed. When solutions do exist, typically they are not unique. Indeed, multiple solutions can arise in a number of ways. For example, some robots are designed with n axes where $n > 6$. For these robots, infinitely many solutions to the inverse kinematics problem typically exist. We refer to robots with more than six axes as *kinematically redundant* robots, because they have more degrees of freedom than are necessary to establish arbitrary tool configurations. These extra degrees of

freedom add flexibility to the manipulator. For example, a redundant robot might be commanded to reach around an obstacle and manipulate an otherwise inaccessible object. Here some of the degrees of freedom can be used to avoid the obstacle while the remaining degrees of freedom are used to configure the tool. Consider, for example, the top view of a redundant SCARA robot shown in Fig. 3-3. Since only two joints are needed to establish the horizontal position of the tool, the second elbow joint of the SCARA robot is redundant.

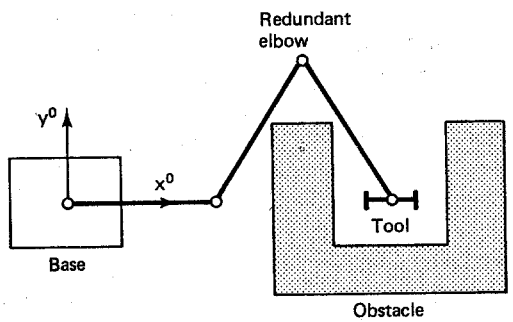


Figure 3-3 Reaching around an obstacle with a redundant robot.

Even when a robot is not kinematically redundant, there are often circumstances in which the solution to the inverse kinematics problem is not unique. Several distinct solutions can arise when the size of the joint-space work envelope Q is sufficiently large. As a case in point, consider the articulated-coordinate robot shown in Fig. 3-4. If the limits on the range of travel for the shoulder, elbow, and tool pitch joints are sufficiently large, then two distinct solutions exist for the simple task of placing the tool out in front of the robot.

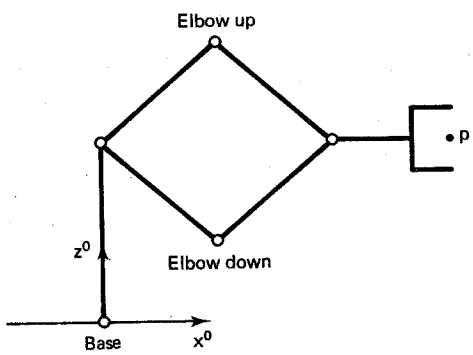


Figure 3-4 Multiple solutions with a nonredundant robot.

We refer to the two solutions in Fig. 3-4 as the *elbow-up* and the *elbow-down* solution. In tool-configuration space the two solutions are identical, because they produce the same p and R, but in joint space they are clearly distinct. Typically the elbow-up solution is preferred, because it reduces the chance of a collision between the links of the arm and obstacles resting on the work surface.

3-3 TOOL CONFIGURATION

In order to develop a solution to the inverse kinematics problem, the desired tool configuration must be specified as input data. Thus far we have assumed that the tool configuration is represented by the pair $\{p, R\}$, where p represents the tool position relative to the base and R represents the tool orientation relative to the base. Specifying tool-tip position with a translation vector p is a natural and convenient technique. However, specifying tool orientation with a rotation matrix R is, at best, awkward, because two-thirds of the information that must be provided is redundant. In this section we introduce an alternative representation of tool configuration that is more compact.

3-3-1 Tool-Configuration Vector

Consider the tool orientation information provided by the approach vector or last column of R. The approach vector effectively specifies both the tool yaw angle and the tool pitch angle, but not the tool roll angle, because the roll angle represents a rotation *about* the approach vector. Here, for convenience, the terms *yaw, pitch,* and *roll* are being used to refer to rotations of the tool about the normal vector, sliding vector, and approach vector, respectively. It would be more precise to refer to them as *turn, tilt,* and *twist* angles (Whitney, 1972).

Given that the approach vector specifies the tool orientation except for the tool roll angle, some method must be found to augment the approach vector with the roll angle information. If we simply append the roll angle, this yields a total of four components, which is not a minimal representation of tool orientation. Instead, note from Eq. (3-2-5) that the approach vector r^3 is a *unit* vector specifying a *direction* only. The size or length of the approach vector can be scaled by a positive quantity without changing the specified direction. This is the key to encoding the tool roll information into the approach vector to produce a minimal representation of orientation.

To recover the tool roll angle from a scaled approach vector, we must use an invertible function of the roll angle q_n to scale the length of r^3. The range of travel of the tool roll joint is specified in the joint-space work envelope Q in Eq. (3-1-2). Often this range is bounded because of the presence of cables running from the wrist to sensors mounted at the tool tip. However, even when q_n is unbounded, the following positive, invertible, exponential *scaling function* can be used:

$$f(q_n) \triangleq \exp \frac{q_n}{\pi} \tag{3-3-1}$$

Other scaling functions might also be used, but the function f is sufficient because $f(q_n) > 0$ for all q_n and $f^{-1}(q_n)$ is well defined. If we scale the approach vector r^3 by $f(q_n)$, this yields a compact representation of tool orientation. We then combine this representation of orientation with the tool tip position and refer to the resulting vector in $\mathbf{R}^6$ as a tool-configuration vector.

Definition 3-3-1: Tool-Configuration Vector. Let p and R denote the position and orientation of the tool frame relative to the base frame where q_n represents the tool roll angle. Then the *tool-configuration vector* is a vector w in $\mathbf{R}^6$ defined:

$$w \triangleq \begin{bmatrix} w^1 \\ \hline w^2 \end{bmatrix} \triangleq \begin{bmatrix} p \\ \hline [\exp(q_n/\pi)]r^3 \end{bmatrix}$$

Since the tool-configuration vector has only six components, it is a *minimal* representation of tool configuration in the general case. The first three components, $w^1 = p$, represent the tool-tip position, while the last three components, $w^2 = [\exp(q_n/\pi)]r^3$, represent the tool orientation. The tool roll angle can be easily recovered from the tool-configuration vector w, as can be seen from the following exercise:

Exercise 3-3-1: Tool Roll. Show that the tool roll angle q_n can be obtained from the tool-configuration vector w as follows:

$$q_n = \pi \ln (w_4^2 + w_5^2 + w_6^2)^{1/2}$$

The tool-configuration vector w provides us with a more convenient way to specify the desired tool position and orientation as input data to the inverse kinematics problem. Furthermore, the *structure* of the tool-configuration vector can reveal useful qualitative information about the types of positions and orientations achievable by the tool. To see this, we examine a five-axis articulated robot.

3-3-2 Tool Configuration of a Five-Axis Articulated Robot

Many industrial robots have five axes; some have only four. Even though these manipulators are not general in the sense of being able to generate arbitrary tool configurations, they are still quite versatile. However, when robotic work cells employing these robots are planned, the layout of the part feeders and other fixtures must be constrained so they do not require the flexibility of a full six-axis robot. As an illustration, consider a five-axis articulated-coordinate robot which has no tool yaw motion. In this case, the part feeders might be arranged concentrically around the robot base as shown in Fig. 3-5.

The part feeders can be tilted at any angle γ that is consistent with the range of global tool pitch angles realizable by the robot. However, the part feeders must be oriented in such a way as to be aligned with radial lines or spokes emanating from the base axis. This is because with the tool yaw angle fixed as in Fig. 3-5, the approach vector of the tool is constrained to lie in a vertical plane through the base axis. This constraint can be summarized algebraically in terms of the components of the arm matrix as follows:

$$R_{13}p_2 = R_{23}p_1 \tag{3-3-2}$$

Here the ratio of the first two components of the approach vector r^3 must equal the ratio of the first two components of the position vector p. In geometric terms, the

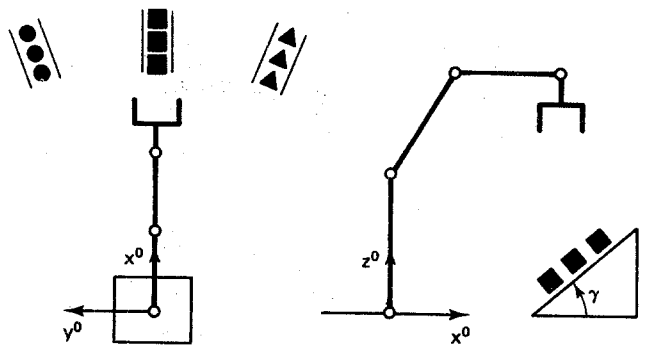


Figure 3-5 Part presentation for a five-axis articulated robot.

tool can manipulate objects from above, from the front, from the back, and even from below consistent with work-envelope constraints and limits on the range of tool pitch angles. However, the five-axis robot shown in Fig. 3-5 cannot manipulate objects from the side. That is, the orthogonal projection of the approach vector onto the work-surface or x^0y^0 plane cannot have any component that is tangent to a circle of radius $\| p \|$ centered at the base axis. Another way to view the constraint on tool orientation is to substitute Eq. (3-3-2) in the general expression for the tool-configuration vector in Def. 3-1-1. This yields the following locus of tool configurations, where β is a constant:

$$w = [w_1, w_2, w_3, \beta w_1, \beta w_2, w_6]^T \tag{3-3-3}$$

Thus, for example, if the tool tip is directly in front of the robot ($w_2 = 0$), the approach vector must point forward ($w_5 = 0$). More generally, the ratio of the horizontal components of the tool-tip position, w_1/w_2, must equal the ratio of the horizontal components of the approach vector, w_4/w_5. It is evident by inspection of the formulation in Eq. (3-3-3) that this robot has only five degrees of freedom, because components w_4 and w_5 cannot be specified independently of w_1 and w_2.

3-3-3 Tool Configurations of a Four-Axis SCARA Robot

The most important special case for manipulating parts corresponds to approaching parts from directly above. This includes picking up a part from a horizontal work surface and placing it down on a horizontal work surface. As an example of a robot designed for these types of operations, consider the four-axis horizontal-jointed or SCARA robot shown in Fig. 3-6. Here the tool yaw and tool pitch angles are fixed in such a way that the tool always points straight down.

Recall that the five-axis robot of Fig. 3-5 had an approach vector r^3 that was restricted to lie in a vertical plane through the base axis. The geometry of a SCARA robot removes one more degree of freedom and restricts the approach vector to lie along a vertical line within that plane, namely:

$$r^3 = -i^3 \tag{3-3-4}$$

The four-axis SCARA robot in Fig. 3-6 is a minimal configuration for a general-purpose robot. The first three axes are the major axes used to position the tool tip, while the fourth axis is a minor axis used to orient the tool by varying the slid-

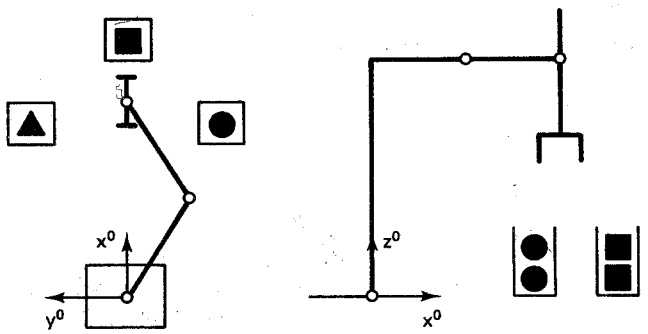


Figure 3-6 Part presentation for a four-axis SCARA robot.

ing vector. If Eq. (3-3-4) is substituted in the general expression for the tool-configuration vector in Def. 3-3-1, this yields:

$$w = \left[p_1,\ p_2,\ p_3,\ 0,\ 0,\ -\exp\frac{q_4}{\pi}\right]^T \tag{3-3-5}$$

The four degrees of freedom that are available are evident from inspection of Eq. (3-3-5), where we see that two of the components of the tool-configuration vector are always zero. The SCARA robot is a particularly simple robot in terms of analysis, yet it is a practical robot for many applications.

3-4 INVERSE KINEMATICS OF A FIVE-AXIS ARTICULATED ROBOT (RHINO XR-3)

The solution to the inverse kinematics problem can be approached either numerically or analytically. The numerical approaches (Uicker et al., 1964) employ search techniques to invert the *tool-configuration function* $w(q)$ by solving the following optimization problem, where $\hat{w} \in \mathbf{R}^6$ is the desired value for the tool configuration:

$$\text{Minimize: } v(q) \triangleq \|w(q) - \hat{w}\|$$

$$\text{subject to: } q \in Q$$

If a $\hat{q} \in Q$ can be found such that $v(\hat{q}) = 0$, then clearly $\hat{q}$ is a solution to the inverse kinematics problem. To solve the optimization problem, a sequence of increasingly accurate approximate solutions $\{q^0, q^1, \ldots\}$ is constructed starting from an initial guess q^0. If the initial guess is sufficiently close to an actual solution, the sequence may converge fairly rapidly. However, for a general q^0, the sequence will converge only to a *local* minimum of $v(q)$, a minimum for which $v(q)$ may or may not be zero. The virtue of the numerical approach lies in the fact that it is *general,* in the sense that the same basic algorithm can be applied to many different manipulators; only $w(q)$ and Q vary. However, the drawbacks are that it is a comparatively slow technique, and furthermore it produces only a single, approximate solution for each initial guess q^0.

The second basic approach to solving the inverse kinematics problem is to exploit the specific nature of the direct kinematic equations to determine an analytical or closed-form expression for the solution. These exact methods are considerably faster than the numerical approximation methods, and they can be used to identify multiple solutions. The main drawback of the analytical methods is that they are robot-dependent. Consequently a separate analysis has to be performed for each robot or generic class of robots (Pieper, 1968).

We illustrate the analytical solution technique by applying it to several different types of robots. The first class of robots we consider are the five-axis vertically-jointed or articulated robots with tool pitch and tool roll motion. This class of robots includes the Rhino XR-3 robot, the Microbot Alpha II robot, and several other commercial robotic manipulators as special cases. The link-coordinate diagram for a five-axis articulated robot was previously developed as Fig. 2-23 in Chap. 2. For convenient reference, we redisplay it here as Fig. 3-7.

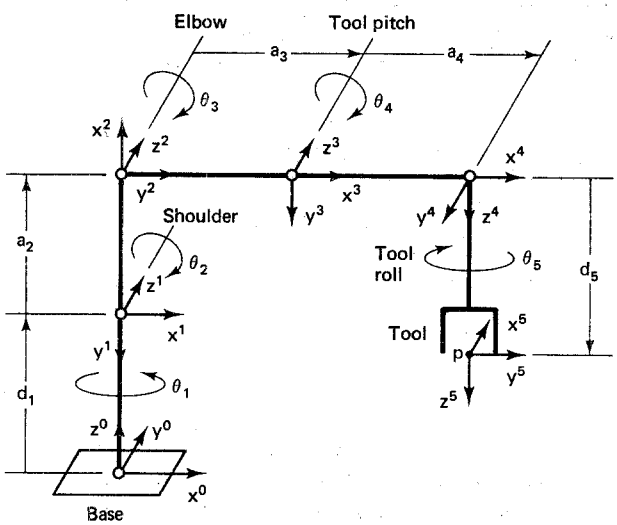


Figure 3-7 Link coordinates of a five-axis articulated robot (Rhino XR-3).

The solution to the inverse kinematics problem starts with the expression for the tool-configuration vector $w(q)$, which can be obtained from the arm matrix developed in Chap. 2. From Prop. 2-7-1 and Def. 3-3-1, the tool-configuration vector for the five-axis articulated arm is:

$$w(q) = \begin{bmatrix} C_1(a_2C_2 + a_3C_{23} + a_4C_{234} - d_5S_{234}) \\ S_1(a_2C_2 + a_3C_{23} + a_4C_{234} - d_5S_{234}) \\ d_1 - a_2S_2 - a_3S_{23} - a_4S_{234} - d_5C_{234} \\ \hline -[\exp(q_5/\pi)]C_1S_{234} \\ -[\exp(q_5/\pi)]S_1S_{234} \\ -[\exp(q_5/\pi)]C_{234} \end{bmatrix} \tag{3-4-1}$$

Since this is an articulated robot, $q = \theta$. Recall that the notation C_{ijk} is short for $\cos(q_i + q_j + q_k)$ and similarly S_{ijk} denotes $\sin(q_i + q_j + q_k)$. In order to extract the joint variable vector q from the nonlinear tool-configuration function $w(q)$,

we apply row operations to the components of w and exploit various trigonometric identities from Appendix 1 in order to isolate the individual joint variables.

3-4-1 Base Joint

Since there is no yaw motion, the easiest joint variable to extract is the base angle q_1. Inspection of the expressions for w_1 and w_2 in Eq. (3-4-1) reveals that they have a factor in common. If we divide w_2 by w_1, this factor cancels, and we are left with S_1/C_1. Thus the base angle is simply:

$$q_1 = \text{atan2}\ (w_2, w_1) \tag{3-4-2}$$

The function atan2 in Eq. (3-4-2) denotes a four-quadrant version of the arctan function. The atan2 function allows us to recover angles over the *entire* range $[-\pi, \pi]$. This function is used repeatedly in subsequent derivations. It can be implemented with the more common two-quadrant arctan function as indicated in Table 3-1, where the notation sgn represents the signum, or sign, function. That is, sgn (α) returns -1 or 1 depending upon whether α is negative or nonnegative, respectively.

TABLE 3-1 FOUR-QUADRANT ARCTAN FUNCTION

Case	Quadrants	atan2 (y, x)
$x > 0$	1, 4	arctan (y/x)
$x = 0$	1, 4	[sgn (y)] $\pi/2$
$x < 0$	2, 3	arctan (y/x) + [sgn (y)] π

3-4-2 Elbow Joint

The elbow angle q_3 is the most difficult joint variable to extract, because it is strongly coupled with the shoulder and tool pitch angles in a vertical-jointed robot. We begin by isolating an intermediate variable, q_{234}, called the *global tool pitch* angle. Here $q_{234} = q_2 + q_3 + q_4$ is the tool pitch angle measured relative to the work surface or x^0y^0 plane. Inspection of the last three components of w in Eq. (3-4-1) reveals that $-(C_1w_4 + S_1w_5)/(-w_6) = S_{234}/C_{234}$. Since the base angle q_1 is already known, the global tool pitch angle can then be computed using:

$$q_{234} = \text{atan2}\ [-(C_1w_4 + S_1w_5), -w_6] \tag{3-4-3}$$

Once the shoulder angle q_2 and elbow angle q_3 are known, the tool pitch angle q_4 can be computed from the global tool pitch angle q_{234}. In order to isolate the shoulder and elbow angles, we define the following two intermediate variables:

$$b_1 \triangleq C_1w_1 + S_1w_2 - a_4C_{234} + d_5S_{234} \tag{3-4-4}$$

$$b_2 \triangleq d_1 - a_4S_{234} - d_5C_{234} - w_3 \tag{3-4-5}$$

Note that b_1 and b_2 are constants whose values are known at this point because q_1 and q_{234} have already been determined. If we take the expressions for the components of w in Eq. (3-4-1) and substitute them in the expressions for b_1 and b_2, this

yields:

$$b_1 = a_2C_2 + a_3C_{23} \tag{3-4-6}$$

$$b_2 = a_2S_2 + a_3S_{23} \tag{3-4-7}$$

We are now left with two independent expressions involving the shoulder and elbow angles; the coupling with the tool pitch angle has been removed. The elbow angle can be isolated by computing $\|b\|^2$. Using trigonometric identities from Appendix 1, we find, after some simplification, that:

$$\|b\|^2 = a_2^2 + 2a_2a_3C_3 + a_3^2 \tag{3-4-8}$$

Note that $\|b\|$ depends only on the elbow angle q_3 because $\|b\|$ represents the distance between the shoulder frame L_1 and the wrist frame L_3. If we now solve Eq. (3-4-8) to recover q_3, this results in two possible solutions, as illustrated previously in Fig. 3-4, the *elbow-up* solution, and *elbow-down* solution, respectively.

$$q_3 = \pm\arccos\frac{\|b\|^2 - a_2^2 - a_3^2}{2a_2a_3} \tag{3-4-9}$$

In view of Eq. (3-4-9), the solution to the inverse kinematics problem for a five-axis articulated-coordinate robot is not unique. In most instances, we use the elbow-up solution, because this keeps the elbow joint further away from the work surface. Indeed, for some commercial robots, this is the only solution allowed, because of joint limit restrictions contained in Q.

3-4-3 Shoulder Joint

To isolate the shoulder angle q_2, we return to the expressions in Eqs. (3-4-6) and (3-4-7), which represent b_1 and b_2 in terms of the shoulder and elbow angles. If we expand C_{23} and S_{23}, using the cosine of the sum and sine of the sum trigonometric identities, and rearrange the terms, this yields:

$$b_1 = (a_2 + a_3C_3)C_2 - (a_3S_3)S_2 \tag{3-4-10}$$

$$b_2 = (a_2 + a_3C_3)S_2 + (a_3S_3)C_2 \tag{3-4-11}$$

Since the elbow angle q_3 is already known, Eqs. (3-4-10) and (3-4-11) constitute a system of two simultaneous linear equations in the unknowns C_2 and S_2. If we use row operations to solve this linear system, the result is:

$$C_2 = \frac{(a_2 + a_3C_3)b_1 + a_3S_3b_2}{\|b\|^2} \tag{3-4-12}$$

$$S_2 = \frac{(a_2 + a_3C_3)b_2 - a_3S_3b_1}{\|b\|^2} \tag{3-4-13}$$

Since we have expressions for both the cosine and the sine of the shoulder angle, we can now recover the shoulder angle over the complete range $[-\pi, \pi]$ using the atan2 function. In particular:

$$q_2 = \text{atan2}\,[(a_2 + a_3C_3)b_2 - a_3S_3b_1, (a_2 + a_3C_3)b_1 + a_3S_3b_2] \tag{3-4-14}$$

3-4-4 Tool Pitch Joint

The work for extracting the tool pitch angle q_4 is already in place. We know the shoulder angle q_2, the elbow angle q_3, and the global tool pitch angle q_{234}. Thus:

$$q_4 = q_{234} - q_2 - q_3 \tag{3-4-15}$$

3-4-5 Tool Roll Joint q_5

The final joint variable is q_5, the tool roll angle. This can be recovered from the last three components of w, as indicated previously in Exercise 3-3-1. In this case, we have:

$$q_5 = \pi \ln (w_4^2 + w_5^2 + w_6^2)^{1/2} \tag{3-4-16}$$

Recall that the tool-configuration vector w is constructed from the position vector p and a scaled version of the approach vector r^3. In certain cases, it may be useful to specify the rotation matrix R and then compute the tool roll angle q_5 from it. To see how this might be done for the five-axis articulated robot, recall from Prop. 2-7-1 that the rotation matrix is:

$$R(q) = \begin{bmatrix} C_1C_{234}C_5 + S_1S_5 & -C_1C_{234}S_5 + S_1C_5 & -C_1S_{234} \\ S_1C_{234}C_5 - C_1S_5 & -S_1C_{234}S_5 - C_1C_5 & -S_1S_{234} \\ -S_{234}C_5 & S_{234}S_5 & -C_{234} \end{bmatrix} \tag{3-4-17}$$

On the surface, it appears that the easiest way to recover q_5 from R is to use the last row of R, where $R_{32}/(-R_{31}) = S_5/C_5$. However, the ratio R_{31}/R_{32} is undefined whenever the global tool pitch angle q_{234} is a multiple of π. Rather than pursue this approach further by decomposing it into special cases, we instead consider the first two components of the normal vector r^1. Taking S_1 times the first component and subtracting C_1 times the second component, the result is:

$$S_5 = S_1R_{11} - C_1R_{21} \tag{3-4-18}$$

From Eq. (3-4-18) we can compute q_5 over the range $[-\pi/2, \pi/2]$. Since we want a general expression for the tool roll angle, one that is valid over the entire range $[-\pi, \pi]$, we also need to determine C_5. Here we consider the first two components of the sliding vector r^2. Taking S_1 times the first component and subtracting C_1 time the second component, the result is:

$$C_5 = S_1R_{12} - C_1R_{22} \tag{3-4-19}$$

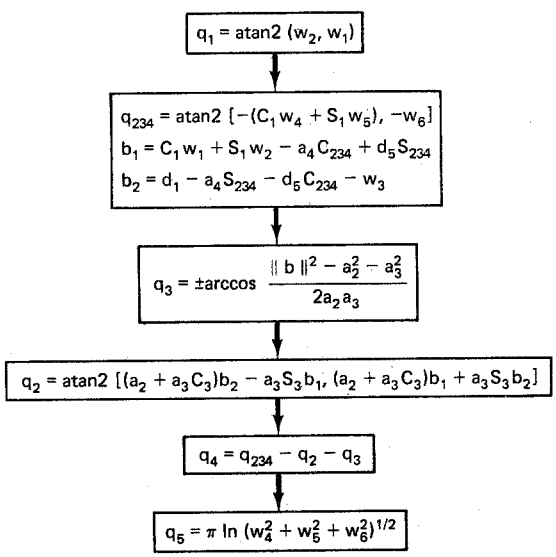
Given both the cosine and the sine of the tool roll angle, we can now evaluate q_5 over the complete range $[-\pi, \pi]$ using the atan2 function. In particular, we have:

$$q_5 = \text{atan2}\,(S_1R_{11} - C_1R_{21}, S_1R_{12} - C_1R_{22}) \tag{3-4-20}$$

3-4-6 Complete Solution

The complete inverse kinematics solution for a five-axis articulated robot with tool pitch and tool roll motion is summarized in Algorithm 3-4-1. Note that this solution applies to both the Rhino XR-3 robot and the Alpha II robot as long as appropriate values for the joint distance vector d and link length vector a are used. In either case, the parameter d_5 represents the tool length, which may vary depending upon which of the various types of optional fingers are used. The two expressions for q_3 specified by the plus and minus signs represent the elbow-up and elbow-down solutions, respectively.

Algorithm 3-4-1: Inverse Kinematics of a Five-Axis Articulated Robot (Rhino XR-3)


$q_1 = \text{atan2}(w_2, w_1)$
$q_{234} = \text{atan2}[-(C_1 w_4 + S_1 w_5), -w_6]$
$b_1 = C_1 w_1 + S_1 w_2 - a_4 C_{234} + d_5 S_{234}$
$b_2 = d_1 - a_4 S_{234} - d_5 C_{234} - w_3$
$q_3 = \pm\arccos \dfrac{\| b \|^2 - a_2^2 - a_3^2}{2a_2 a_3}$
$q_2 = \text{atan2}[(a_2 + a_3 C_3)b_2 - a_3 S_3 b_1, (a_2 + a_3 C_3)b_1 + a_3 S_3 b_2]$
$q_4 = q_{234} - q_2 - q_3$
$q_5 = \pi \ln (w_4^2 + w_5^2 + w_6^2)^{1/2}$

The solution to the inverse kinematics problem outlined in Algorithm 3-4-1 assumes that the tool-configuration vector w is supplied as input data. If instead the tool position and orientation $\{p, R\}$ are supplied directly, then $w = [p_1, p_2, p_3, R_{13}, R_{23}, R_{33}]^T$ can be used to compute $\{q_1, q_2, q_3, q_4\}$ from Algorithm 3-4-1 and q_5 can be computed from R using Eq. (3-4-20). A third possibility is to use something called a *reduced tool-configuration vector* $\tilde{w}$ as input to the inverse kinematics algorithm. For an n-axis robot with $n < 6$, a reduced tool-configuration vector is a vector in $\mathbf{R}^n$ rather than $\mathbf{R}^6$. This way each of the components of $\tilde{w}$ can be independently specified. In the case of a five-axis articulated robot with tool pitch and tool roll motion, the following is an example of a reduced tool-configuration vector

$$\tilde{w} = [p_1, p_2, p_3, q_{234}, q_{1-5}]^T \tag{3-4-21}$$

Here the tool-tip position p is supplied and two parameters describing tool orientation are added. The first is the global tool pitch angle $q_{234} = q_2 + q_3 + q_4$, which is the tool pitch measured relative to the work surface or x^0y^0 plane. The second is the global tool roll angle $q_{1-5} = q_1 - q_5$, which is the tool roll measured relative to x^0 when $r^3 = -i^3$. If $\tilde{w}$ is used as input data, then Algorithm 3-4-1 simplifies with $q_{234} = \tilde{w}_4$ and $q_5 = q_1 - \tilde{w}_5$.

Exercise 3-4-1: Home Position. For the five-axis articulated robot in Fig. 3-7, use Algorithm 3-4-1 to find q when the robot is in the following position, called the *home position*. Does your answer agree with the last column of Table 2-4?

$$w = [a_3 + a_4,\ 0,\ d_1 + a_2 - d_5,\ 0,\ 0,\ -0.607]^T$$

Exercise 3-4-2: Zero Position. For the five-axis articulated robot in Fig. 3-7, use Algorithm 3-4-1 to find q when the robot is in the following position, called the *zero position:*

$$w = [a_2 + a_3 + a_4,\ 0,\ d_1 - d_5,\ 0,\ 0,\ -1]^T$$

Exercise 3-4-3: Maximum Reach. For the five-axis articulated robot in Fig. 3-7, use Algorithm 3-4-1 to find q when the robot is in the following positions:

1. A maximum horizontal reach position
2. A maximum vertical reach position

3-5 INVERSE KINEMATICS OF A FOUR-AXIS SCARA ROBOT (ADEPT ONE)

Another important class of robotic manipulators for which a generic solution to the inverse kinematics problem can be obtained is the four-axis horizontal-jointed robot, or SCARA robot. Examples of robots which belong to this class include the Intelledex 440 robot, the IBM 7545 robot, and the Adept One robot. The link-coordinate diagram for a four-axis SCARA robot was previously developed as Fig. 2-25 in Chap. 2. For convenient reference, we redisplay it as Fig. 3-8.

Unlike the vertical-jointed robot, the vector of joint variables in this case is not θ but is instead $q = [\theta_1,\ \theta_2,\ d_3,\ \theta_4]^T$. The arm matrix for the SCARA robot has been previously computed in Chap. 2. From Prop. 2-8-1 and Def. 3-3-1, we get the following expression for the tool-configuration vector for the SCARA robot:

$$w(q) = \begin{bmatrix} a_1C_1 + a_2C_{1-2} \\ a_1S_1 + a_2S_{1-2} \\ d_1 - q_3 - d_4 \\ \hline 0 \\ 0 \\ -\exp\,(q_4/\pi) \end{bmatrix} \tag{3-5-1}$$

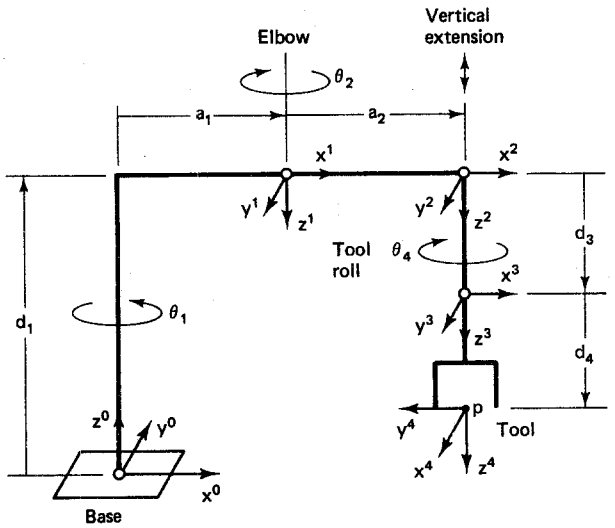


Figure 3-8 Link coordinates of a four-axis SCARA robot (Adept One).

Recall that C_{1-2} is short for cos $(q_1 - q_2)$ and S_{1-2} denotes sin $(q_1 - q_2)$. Note from the last three components of w that the approach vector r^3 is *fixed* at $r^3 = -i^3$. This is characteristic of SCARA-type robots in general, which always approach parts directly from above. In order to extract the joint angle vector q from the nonlinear tool-configuration function $w(q)$, we again apply row operations to the components of w and exploit trigonometric identities from Appendix 1 in order to isolate the individual joint variables.

3-5-1 Elbow Joint

We can extract the elbow angle q_2 from the first two components of the tool-configuration vector w. Note that $w_1^2 + w_2^2$ represents the square of the radial component of the tool position, a quantity which should be independent of the base angle q_1. If we compute $w_1^2 + w_2^2$ from Eq. (3-5-1), then after simplification using trigonometric identities the result is:

$$w_1^2 + w_2^2 = a_1^2 + 2a_1a_2C_2 + a_2^2 \tag{3-5-2}$$

Thus the cosine of the elbow angle has been determined in terms of known constants. If we now solve Eq. (3-5-2) to recover q_2, this results in two possible solutions. The first one is called the *left-handed* solution, and the second one is the *right-handed* solution:

$$q_2 = \pm\arccos\frac{w_1^2 + w_2^2 - a_1^2 - a_2^2}{2a_1a_2} \tag{3-5-3}$$

It follows from Eq. (3-5-3) that the solution to the inverse kinematics problem for a four-axis SCARA robot is not unique. The motivation for the terms *left-handed* and *right-handed* can be seen in Fig. 3-8. If the robot is viewed from above, then, when $q_2 > 0$, the robotic arm curls in from the left, whereas when $q_2 < 0$, it curls in from the right.

3-5-2 Base Joint

The base angle q_1, which might also be thought of as a horizontal shoulder angle, can now be obtained from q_2. First we take the expressions for w_1 and w_2 in Eq. (3-5-1), and expand the C_{1-2} and S_{1-2} terms using the cosine of the difference and sine of the difference trigonometric identities in Appendix 1. After rearranging the coefficients, this yields:

$$(a_1 + a_2C_2)C_1 + (a_2S_2)S_1 = w_1 \tag{3-5-4}$$

$$(-a_2S_2)C_1 + (a_1 + a_2C_2)S_1 = w_2 \tag{3-5-5}$$

Since the elbow angle q_2 is already known, this is a system of two simultaneous linear equations in the unknowns C_1 and S_1. If we use row operations to solve this linear system, the result is:

$$S_1 = \frac{a_2S_2w_1 + (a_1 + a_2C_2)w_2}{(a_2S_2)^2 + (a_1 + a_2C_2)^2} \tag{3-5-6}$$

$$C_1 = \frac{(a_1 + a_2C_2)w_1 - a_2S_2w_2}{(a_2S_2)^2 + (a_1 + a_2C_2)^2} \tag{3-5-7}$$

Since we have expressions for both the cosine and the sine of the base angle, we can again recover the base angle over the complete range $[-\pi, \pi]$ using the atan2 function:

$$q_1 = \text{atan2}\,[a_2S_2w_1 + (a_1 + a_2C_2)w_2, (a_1 + a_2C_2)w_1 - a_2S_2w_2] \tag{3-5-8}$$

3-5-3 Vertical Extension Joint

The prismatic joint variable q_3 is associated with sliding the tool up and down along the tool roll axis. In a SCARA-type robot, the vertical component of the tool motion is uncoupled from the horizontal component, as can be seen from the expression for w in Eq. (3-5-1). Extracting the prismatic joint variable q_3 is therefore a simple matter:

$$q_3 = d_1 - d_4 - w_3 \tag{3-5-9}$$

3-5-4 Tool Roll Joint

The final joint variable is the tool roll angle q_4. Inspection of the last component of w reveals that q_4 can be easily recovered from it as follows:

$$q_4 = \pi \ln |w_6| \tag{3-5-10}$$

As an alternataive to specifying w, one could instead specify the rotation matrix R and then compute the tool roll angle q_4 from it. To see how this might be done for the four-axis SCARA robot, recall from Prop. 2-8-1 that the rotation matrix is:

$$R(q) = \begin{bmatrix} C_{1-2-4} & S_{1-2-4} & 0 \\ S_{1-2-4} & -C_{1-2-4} & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{3-5-11}$$

The angle q_{1-2-4} is called the *global tool roll* angle. It is the tool roll angle measured relative to the base frame of the robot. Note from the normal vector in Eq. (3-5-11) that $R_{21}/R_{11} = S_{1-2-4}/C_{1-2-4}$. Thus the global tool roll angle can be computed from R as follows:

$$q_{1-2-4} = \text{atan2}\,(R_{21}, R_{11}) \tag{3-5-12}$$

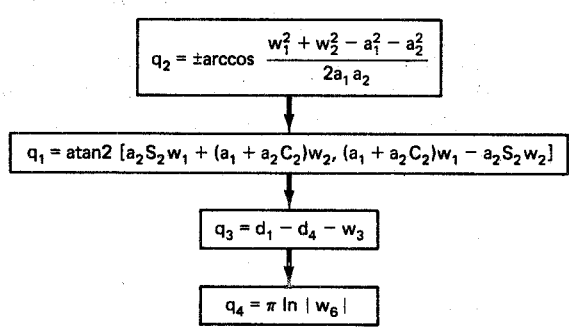
Once the global tool roll angle is known, the tool roll angle q_4 can be computed directly from it because q_1 and q_2 are also known:

$$q_4 = q_1 - q_2 - q_{1-2-4} \tag{3-5-13}$$

3-5-5 Complete Solution

The complete inverse kinematics solution for a four-axis SCARA-type robot is summarized in Algorithm 3-5-1. The fixed joint distances d and the link lengths a will depend on the particular robot. These parameters can be obtained from specification sheets supplied by the robot manufacturer. The two expressions for q_2 specified by the plus and minus signs represent the left-handed and right-handed solutions, respectively.

Algorithm 3-5-1: Inverse Kinematics of a Four-Axis SCARA Robot (Adept One)



The solution to the inverse kinematics problem outlined in Algorithm 3-5-1 is based on the assumption that the tool-configuration vector w is supplied as input data. As an alternative, the pair $\{p, R\}$ can be specified. In this case

$w = [p_1, p_1, p_3, R_{13}, R_{23}, R_{33}]^T$ can be used to compute $\{q_1, q_2, q_3\}$ from Algorithm 3-5-1 and then q_4 can be computed from R using Eqs. (3-5-12) and (3-5-13). A third possibility is to again use a reduced tool-configuration vector $\tilde{w}$ as input to the inverse kinematics algorithm. In the case of a four-axis SCARA robot, the following vector in $\mathbf{R}^4$ is an example of a reduced tool-configuration vector:

$$\tilde{w} = [p_1, p_2, p_3, q_{1-2-4}]^T \tag{3-5-14}$$

Here the tool-tip position p is supplied and one additional parameter describing tool orientation is added. It is the global tool roll angle q_{1-2-4} which is the tool roll measured relative to the base frame. When $\tilde{w}_4 = 0$, the x axes of the tool and the base point in the same direction and positive values for $\tilde{w}_4$ correspond to counterclockwise tool rotations as seen from above. If the reduced tool-configuration vector is used as input data, Algorithm 3-5-1 can be used to compute $\{q_1, q_2, q_3\}$, and q_4 is computed from Eq. (3-5-13) with $q_{1-2-4} = \tilde{w}_4$.

Exercise 3-5-1: Minimum Reach. For the four-axis SCARA robot, suppose the upper arm and the forearm are of equal length ($a_2 = a_1$). Find the joint variables q when the tool tip reaches back and just touches the base axis, as follows:

$$w = [0, 0, 0, 0, 0, -1]^T$$

Exercise 3-5-2: Maximum Reach. For the four-axis SCARA robot, find the joint variables q when the radial extension of the arm is maximum, as follows:

$$w = [a_1 + a_2, 0, 0, 0, 0, -1]^T$$

3-6 INVERSE KINEMATICS OF A SIX-AXIS ARTICULATED ROBOT (INTELLEDEX 660)

As a third example of computing an inverse kinematic solution, consider the six-axis Intelledex 660 articulated robot. The link-coordinate diagram for the Intelledex 660 was previously developed as Fig. 2-27 in Chap. 2. For convenient reference, it is redisplayed here as Fig. 3-9.

Since this is an articulated robot, $q = \theta$. The solution to the inverse kinematics problem starts with the expression for the tool-configuration vector $w(q)$, which can be obtained from the arm matrix developed in Chap. 2. From Prop. 2-9-1 and Def. 3-3-1, the tool-configuration vector for the Intelledex 660 robotic arm is:

$$w(q) = \begin{bmatrix} C_1C_2(C_3a_3 + C_{34}a_4 + S_{345}d_6) + S_1(S_3a_3 + S_{34}a_4 - C_{345}d_6) \\ S_1C_2(C_3a_3 + C_{34}a_4 + S_{345}d_6) - C_1(S_3a_3 + S_{34}a_4 - C_{345}d_6) \\ d_1 + S_2(C_3a_3 + C_{34}a_4 + S_{345}d_6) \\ \hline [\exp(q_6/\pi)](-S_1C_{345} + C_1C_2S_{345}) \\ [\exp(q_6/\pi)](C_1C_{345} + S_1C_2S_{345}) \\ [\exp(q_6/\pi)]S_2S_{345} \end{bmatrix} \tag{3-6-1}$$

Recall that the notation C_{ijk} is short for $\cos(q_i + q_j + q_k)$ and, similarly, S_{ijk}

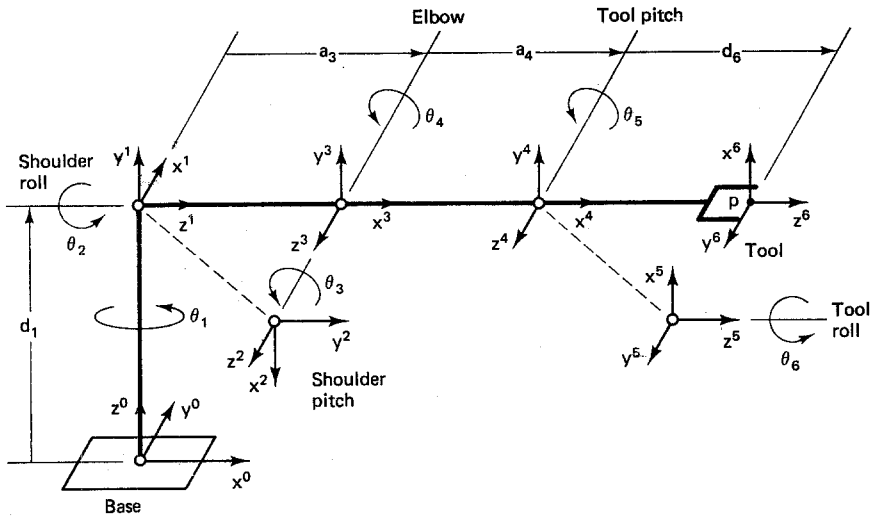


Figure 3-9 Link coordinates of a six-axis articulated robot (Intelledex 660T).

denotes sin $(q_i + q_j + q_k)$. We also make use of the expression for the rotation matrix $R(q)$ developed in Prop. 2-9-1:

$$R = \left[\begin{array}{c|c|c} (C_1C_2C_{345} + S_1S_{345})C_6 + C_1S_2S_6 & C_1S_2C_6 - (C_1C_2C_{345} + S_1S_{345})S_6 & -S_1C_{345} + C_1C_2S_{345} \\ (S_1C_2C_{345} - C_1S_{345})C_6 + S_1S_2S_6 & S_1S_2C_6 - (S_1C_2C_{345} - C_1S_{345})S_6 & C_1C_{345} + S_1C_2S_{345} \\ S_2C_{345}C_6 - C_2S_6 & -C_2C_6 - S_2C_{345}S_6 & S_2S_{345} \end{array}\right] \tag{3-6-2}$$

In order to extract the joint variable vector q from the tool-configuration vector $w(q)$ and the rotation matrix $R(q)$, we apply row operations to w and R and exploit a host of trigonometric identities in order to isolate the individual joint variables.

3-6-1 Tool Roll Joint

The Intelledex 660 robot is quite complex, and in order to develop closed-form expressions for the joint variables, we assume that the tool configuration is available both in the form of the tool-configuration vector w and the translation and rotation pair $\{p, R\}$. From Exercise 3-3-1, the tool roll angle q_6 can be recovered from the last three components of the tool-configuration vector w as follows:

$$q_6 = \pi \ln (w_4^2 + w_5^2 + w_6^2)^{1/2} \tag{3-6-3}$$

3-6-2 Shoulder Roll Joint

Given the tool roll angle, the shoulder roll angle q_2 can be computed from the third components of the normal vector r^1 and the sliding vector r^2. If we formulate $R_{31}S_6 + R_{32}C_6$, this causes the common term involving C_{234} to drop out. The re-

mainder, after simplification using $C_6^2 + S_6^2 = 1$, is simply $-C_2$. Consequently, we can solve for q_2 as follows:

$$q_2 = \pm\arccos(-R_{31}S_6 - R_{32}C_6) \tag{3-6-4}$$

Clearly, the solution is not unique. Recall from Table 2-6 that in the home position, $q_2 = -\pi/2$. We therefore choose the negative solution and restrict our attention to angles in the range

$$-\pi < q_2 < 0 \tag{3-6-5}$$

The solution corresponding to $q_2 < 0$ is called the *left-handed* solution, because, from Fig. 2-26, it corresponds to the upper arm being on the left side of the small shoulder and forearm. If the shoulder is rolled by π to produce angles in the range $0 < q_2 < \pi$, this places the arm in the *right-handed* configuration. The end points $\{-\pi, 0\}$ are not included in Eq. (3-6-5), for reasons which will become apparent as we proceed.

3-6-3 Base Joint

To extract the base angle q_1, we make use of the first and second components of the normal vector r^1 and the sliding vector r^2. First we compute the quantity $R_{11}S_6 + R_{22}C_6$. The result, after simplification using the identity $S_6^2 + C_6^2 = 1$, is:

$$R_{11}S_6 + R_{12}C_6 = C_1S_2 \tag{3-6-6}$$

Recall from Eqs. (3-6-4) and (3-6-5) that S_2 is a known nonzero quantity. But q_6 is also known, from Eq. (3-6-3), and hence we have isolated C_1. To recover the base angle over the entire range $[-\pi, \pi]$, we must also isolate S_1. This can be achieved with the second components of the normal and sliding vector. Again using the trigonometric identity $S_6^2 + C_6^2 = 1$, we find:

$$R_{21}S_6 + R_{22}C_6 = S_1S_2 \tag{3-6-7}$$

If we now divide Eq. (3-6-7) by Eq. (3-6-6), the nonzero factor S_2 cancels and we are left with an expression for S_1/C_1. Thus the base angle q_1 over the range $[-\pi, \pi]$ is:

$$q_1 = \text{atan2}(R_{21}S_6 + R_{22}C_6, R_{11}S_6 + R_{12}C_6) \tag{3-6-8}$$

3-6-4 Elbow Joint

Perhaps the most difficult joint angle to extract is the elbow angle q_4, which is tightly bound to the shoulder pitch angle q_3 and the tool pitch angle q_5. We can remove the effects of the tool pitch angle q_5 by partitioning the problem at the wrist, as suggested in Eqs. (3-2-7) and (3-2-8). If we subtract d_6r^3 from the tool tip position p, this yields the wrist position relative to the base. To simplify the subsequent equations, we also subtract d_1i^3 and denote the result as b. That is,

$$b \triangleq p - d_6r^3 - d_1i^3 \tag{3-6-9}$$

The first two components of the intermediate variable b specify the x and y coordinates of the wrist frame L_4 relative to the base frame L_0. The last component of b specifies the vertical distance from the wrist frame L_4 to the shoulder frame L_2. If we substitute for p and r^3 in Eq. (3-6-9), we get the following expressions for the components of b:

$$b_1 = C_1C_2(C_3a_3 + C_{34}a_4) + S_1(S_3a_3 + S_{34}a_4) \tag{3-6-10}$$

$$b_2 = S_1C_2(C_3a_3 + C_{34}a_4) - C_1(S_3a_3 + S_{34}a_4) \tag{3-6-11}$$

$$b_3 = S_2(C_3a_3 + C_{34}a_4) \tag{3-6-12}$$

Note how the effects of the tool pitch angle q_5 have been removed from the original expression for p. The quantity $\|b\|$ represents the straight-line distance from the shoulder joint to the wrist joint. Because the elbow joint is the *only* joint between the shoulder and the wrist, $\|b\|$ should depend exclusively on q_4. If we compute $\|b\|^2$, the result after simplification using various trigonometric identities in Appendix 1 is:

$$\|b\|^2 = a_3^2 + 2a_3a_4C_4 + a_4^2 \tag{3-6-13}$$

Thus $\|b\|$ depends exclusively on q_4, as expected. Solving Eq. (3-6-13) for q_4 then yields:

$$q_4 = \pm\arccos\frac{\|b\|^2 - a_3^2 - a_4^2}{2a_3a_4} \tag{3-6-14}$$

Again we see that the solution is not unique. The positive solution returns angles in the range $[0, \pi]$ and is referred to as the *elbow-down* solution, while the negative solution, which produces angles in the range $[-\pi, 0]$, is the called the *elbow-up* solution.

3-6-5 Shoulder Pitch Joint

The shoulder pitch angle q_3 is also a challenge to extract. The basic approach is to find two independent linear equations in S_3 and C_3. These can be obtained from the expressions for b now that the angles q_1, q_2, and q_4 are known. First, note from Eqs. (3-6-10) and (3-6-11) that $S_1b_1 - C_1b_2 = S_3a_3 + S_{34}a_4$. If we use the sine of the sum trigonometric identity for S_{34} and rearrange the coefficients, this yields:

$$S_1b_1 - C_1b_2 = (a_3 + C_4a_4)S_3 + (S_4a_4)C_3 \tag{3-6-15}$$

To produce a second equation involving the unknowns S_3 and C_3, we note from Eq. (3-6-12) that $b_3/S_2 = C_3a_3 + C_{34}a_4$. If we use the cosine of the sum trigonometric identity for C_{34} and rearrange the coefficients, the result is:

$$\frac{b_3}{S_2} = (-S_4a_4)S_3 + (a_3 + C_4a_4)C_3 \tag{3-6-16}$$

Note that the expression in Eq. (3-6-16) involves division by S_2. This is possible because $S_2 \neq 0$ in view of Eq. (3-6-5). We can now solve Eqs. (3-6-15) and (3-6-16) simultaneously using row operations, and the result is:

$$C_3 = \frac{S_4a_4(S_1b_1 - C_1b_2) + (a_3 + C_4a_4)b_3/S_2}{(S_4a_4)^2 + (a_3 + C_4a_4)^2} \tag{3-6-17}$$

$$S_3 = \frac{(a_3 + C_4a_4)(S_1b_1 - C_1b_2) - S_4a_4b_3/S_2}{(S_4a_4)^2 + (a_3 + C_4a_4)^2} \tag{3-6-18}$$

Given the expressions for C_3 and S_3, the atan2 function can now be used to solve for the shoulder pitch angle over the entire range $[-\pi, \pi]$. This yields:

$$q_3 = \text{atan2}\left[(a_3 + C_4a_4)(S_1b_1 - C_1b_2) - \frac{S_4a_4b_3}{S_2}, \; S_4a_4(S_1b_1 - C_1b_2) + \frac{(a_3 + C_4a_4)b_3}{S_2}\right] \tag{3-6-19}$$

3-6-6 Tool Pitch Joint

The only joint variable that remains to be isolated is the tool pitch angle q_5. We determine q_5 indirectly by first computing the sum angle q_{345} using the R matrix. Observe from Eq. (3-6-2) that $R_{33} = S_2S_{345}$. Because S_2 is known and nonzero, this means that q_{345} is known over the interval $[-\pi/2, \pi/2]$. To extend the solution to the entire interval $[-\pi, \pi]$, we must also isolate C_{345}. This can be achieved by using the remaining components in the third row of R. Note from Eq. (3-6-2) that $R_{31}C_6 - R_{32}S_6 = S_2C_{345}$. It then follows that the sum angle q_{345} is:

$$q_{345} = \text{atan2}\,(R_{33}, R_{31}C_6 - R_{32}S_6) \tag{3-6-20}$$

The work for extracting the final joint variable is now in place because q_3, q_4, and q_{345} are all known. Recalling that $q_{345} = q_3 + q_4 + q_5$, it follows that:

$$q_5 = q_{345} - q_3 - q_4 \tag{3-6-21}$$

3-6-7 Complete Solution

The complete inverse kinematics solution for the six-axis Intelledex 660 robot is summarized in Algorithm 3-6-1. Recall that this solution is valid only for the left-handed configuration with $-\pi < q_2 < 0$. The right-handed solution can be obtained by simply changing the sign of the arccos term in the expression for q_2. However, in either case, Algorithm 3-6-1 is not valid when the shoulder roll angle q_2 is a multiple of π, because it is assumed that $S_2 \neq 0$. This assumption is explicit in the expression for the shoulder pitch angle q_3 and implicit in the expressions for q_1 and q_{345}.

Exercise 3-6-1: Home Position. For the six-axis Intelledex 660 robot, use Algorithm 3-6-1 to find q when the robot is in the following position, called the *home position*. Does your answer agree with the last column of Table 2-6?

$$w = [a_3 + a_4 + d_6, \; 0, \; d_1, \; 1, \; 0, \; 0]^T$$

$$R = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Algorithm 3-6-1: Inverse Kinematics of a Six-Axis Articulated Robot (Intelledex 660)

$$q_6 = \pi \ln (w_4^2 + w_5^2 + w_6^2)^{1/2}$$

$$q_2 = -\arccos (-R_{31} S_6 - R_{32} C_6)$$

$$q_1 = \text{atan2} (R_{21} S_6 + R_{22} C_6, R_{11} S_6 + R_{12} C_6)$$

$$b = p - d_6 r^3 - d_1 i^3$$
$$q_{345} = \text{atan2} (R_{33}, R_{31} C_6 - R_{32} S_6)$$

$$q_4 = \pm \arccos \frac{\| b \|^2 - a_3^2 - a_4^2}{2 a_3 a_4}$$

$$q_3 = \text{atan2} \left[(a_3 + C_4 a_4)(S_1 b_1 - C_1 b_2) - \frac{S_4 a_4 b_3}{S_2}, S_4 a_4 (S_1 b_1 - C_1 b_2) + \frac{(a_3 + C_4 a_4) b_3}{S_2} \right]$$

$$q_5 = q_{345} - q_3 - q_4$$

3-7 INVERSE KINEMATICS OF A THREE-AXIS PLANAR ARTICULATED ROBOT

As a final example of computing the inverse kinematics of a robot, consider the three-axis planar articulated robot shown in Fig. 3-10. For a robot which operates in a plane, there are only three degrees of freedom for the tool: two translational motions within the plane, and one rotational motion about an axis orthogonal to the plane.

To solve the inverse kinematic equations of the three-axis planar manipulator, we must first derive the direct kinematic equations. The application of steps 0 to 7

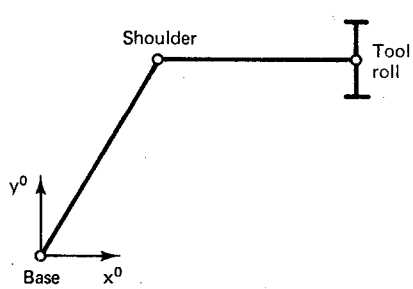


Figure 3-10 A three-axis planar articulated robot.

of the D-H algorithm to the three-axis planar articulated robot yields the link-coordinate diagram shown in Fig. 3-11.

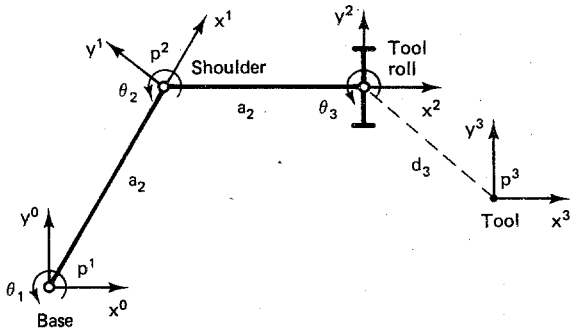


Figure 3-11 Link coordinates of a three-axis planar articulated robot.

Next, the application of steps 8 to 13 of the D-H algorithm results in the kinematic parameters shown in Table 3-2.

TABLE 3-2 KINEMATIC PARAMETERS OF A THREE-AXIS PLANAR ARTICULATED ROBOT

Axis	θ	d	a	α	Home
1	q_1	0	a_1	0	$\pi/3$
2	q_2	0	a_2	0	$-\pi/3$
3	q_3	d_3	0	0	0

Since the robot is an articulated robot, the vector of joint variables is $q = \theta$. Using Table 3-2 and Prop. 2-6-1, the arm matrix for the three-axis articulated robot can be computed as follows:

$$
\begin{aligned}
T_{\text{base}}^{\text{tool}} &= T_0^1 T_1^2 T_2^3 \\
&= \begin{bmatrix} C_1 & -S_1 & 0 & a_1C_1 \\ S_1 & C_1 & 0 & a_1S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_3 & -S_3 & 0 & 0 \\ S_3 & C_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} C_{12} & -S_{12} & 0 & a_1C_1 + a_2C_{12} \\ S_{12} & C_{12} & 0 & a_1S_1 + a_2S_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_3 & -S_3 & 0 & 0 \\ S_3 & C_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} C_{123} & -S_{123} & 0 & a_1C_1 + a_2C_{12} \\ S_{123} & C_{123} & 0 & a_1S_1 + a_2S_{12} \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{3-7-1}
$$

Exercise 3-7-1: Planar *n*-Axis Manipulator. Use the pattern in T_0^1, T_0^2, and T_0^3 to write down, by inspection, the arm matrix of an n-axis planar articulated robot, where n is arbitrary.

At this point we combine the tool-tip position and tool orientation into the more compact tool-configuration vector w. Although the tool-configuration space could be regarded as three-dimensional in this case, to facilitate comparisons with other robots we use the general case $w = (w^1, w^2)$, where $w^1 = p$ and $w^2 = [\exp(q_3/\pi)]r^3$. From inspection of the arm matrix in Eq. (3-7-1), we see that the tool-configuration function of the three-axis planar articulated robot is:

$$w(q) = \begin{bmatrix} a_1C_1 + a_2C_{12} \\ a_1S_1 + a_2S_{12} \\ d_3 \\ \hline 0 \\ 0 \\ \exp(q_3/\pi) \end{bmatrix} \tag{3-7-2}$$

The planar nature of the three-axis robot is evident from Eq. (3-7-2), since $w_3(q)$, $w_4(q)$, and $w_5(q)$ are all constant. If the tool length d_3 is zero, then the tool tip moves in the x^0y^0 plane. More generally, the tool tip p moves in a plane parallel to the x^0y^0 plane at a distance d_3 above it.

In order to extract the joint angle vector q from the nonlinear tool-configuration function $w(q)$, we again apply row operations to the components of w and exploit trigonometric identities from Appendix 1 in order to isolate the individual joint variables.

3-7-1 Shoulder Joint

First, note from Fig. 3-11 that the radial distance between the base frame L_0 and the wrist frame L_2 depends only on the shoulder angle q_2. In particular, from Eq. (3-7-2) we have:

$$\begin{aligned} w_1^2 + w_2^2 &= (a_1C_1 + a_2C_{12})^2 + (a_1S_1 + a_2S_{12})^2 \\ &= a_1^2C_1^2 + 2a_1a_2C_1C_{12} + a_2^2C_{12}^2 + a_1^2S_1^2 + 2a_1a_2S_1S_{12} + a_2^2S_{12}^2 \\ &= a_1^2 + 2a_1a_2(C_{12}C_1 + S_{12}S_1) + a_2^2 \\ &= a_1^2 + 2a_1a_2C_2 + a_2^2 \end{aligned} \tag{3-7-3}$$

We can now solve Eq. (3-7-3) for the shoulder angle. Isolating C_2 and applying the arccos function to both sides yields the following two solutions:

$$q_2 = \pm\arccos\frac{w_1^2 + w_2^2 - a_1^2 - a_2^2}{2a_1a_2} \tag{3-7-4}$$

As was the case with the SCARA robot, the first solution is called the left-handed solution and the second is the right-handed solution.

3-7-2 Base Joint

Given the shoulder angle q_2, the base angle can now be determined from Eq. (3-7-2). If the expressions for C_{12} and S_{12} are expanded using the cosine of the sum and sine of the sum trigonometric identities and the coefficients of C_1 and S_1 are collected, this yields:

$$(a_1 + a_2C_2)C_1 - (a_2S_2)S_1 = w_1 \tag{3-7-5a}$$

$$(a_2S_2)C_1 + (a_1 + a_2C_2)S_1 = w_2 \tag{3-7-5b}$$

Thus we have two linear equations in the unknowns C_1 and S_1. These equations can be solved simultaneously using elementary row operations, and the result is:

$$C_1 = \frac{(a_1 + a_2C_2)w_1 + a_2S_2w_2}{(a_1 + a_2C_2)^2 + (a_2S_2)^2} \tag{3-7-6a}$$

$$S_1 = \frac{(a_1 + a_2C_2)w_2 - a_2S_2w_1}{(a_1 + a_2C_2)^2 + (a_2S_2)^2} \tag{3-7-6b}$$

Note that if $a_2 = a_1$, then the numerators and the denominators of the expressions for C_1 and S_1 go to zero when $q_2 = \pi$. This corresponds to a workspace singularity with the tool roll axis collinear with the base axis. If we *avoid* this singularity, then both C_1 and S_1 are known and the base angle q_1 can be recovered over the complete range $[-\pi, \pi]$ using:

$$q_1 = \text{atan2}\,[(a_1 + a_2C_2)w_2 - a_2S_2w_1, (a_1 + a_2C_2)w_1 + a_2S_2w_2] \tag{3-7-7}$$

3-7-3 Tool Roll Joint

From Eq. (3-7-1), the tool roll angle q_3 can be obtained directly from the last component of the tool-configuration vector as follows:

$$q_3 = \pi \ln w_6 \tag{3-7-8}$$

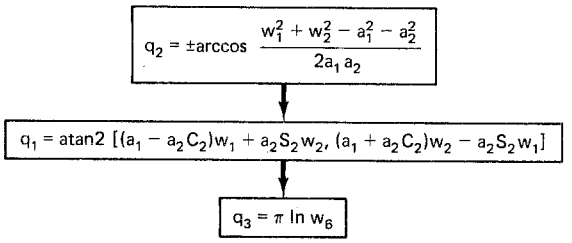
3-7-4 Complete Solution

The complete inverse kinematics solution for the three-axis planar articulated robot in Fig. 3-11 is summarized in Algorithm 3-7-1. Recall that the two expressions for q_2 specified by the plus and minus signs represent the left-handed and right-handed solutions, respectively.

The solution to the inverse kinematics problem outlined in Algorithm 3-7-1 assumes that the tool-configuration vector w is supplied as input data. If instead the pair $\{p, R\}$ is specified, then $w = [p_1, p_1, p_3, R_{13}, R_{23}, R_{33}]^T$ can be used to compute q_1 and q_2 using Algorithm 3-7-1. From Eq. (3-7-2), the tool roll angle can then be computed from the global tool roll angle q_{123}, as follows:

$$q_3 = \text{atan2}\,(R_{11}, R_{21}) - q_1 - q_2 \tag{3-7-9}$$

Algorithm 3-7-1: Inverse Kinematics of a Three-Axis Planar Articulated Robot



Exercise 3-7-2: Right Angle. For the three-axis planar articulated robot in Fig. 3-11, find the joint variables q when the first two links form a right angle, as follows:

$$w = [a_2, a_1, d_3, 0, 0, 1]^T$$

3-8 A ROBOTIC WORK CELL

The technique for transforming between tool coordinates and base coordinates using homogeneous coordinate transformation matrices can also be applied more generally to coordinate transformations between various *stations* in a robotic work cell. As an illustration, consider the single-robot work cell shown in Fig. 3-12. Here a part in the shape of a block has been extracted, say, from a feeder, and placed in the viewing area of an overhead camera. The vision system inspects the part and then informs the robot controller to pick up the part and place it in either a *pass* bin or a *reject* bin, depending upon the outcome of the inspection process.

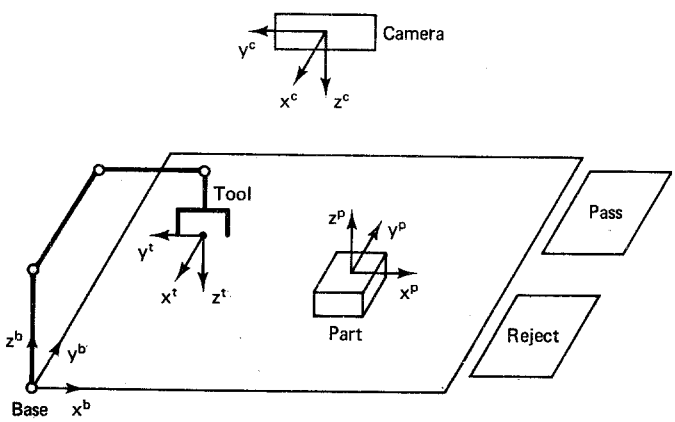


Figure 3-12 A single-robot work cell.

In order to perform a kinematic analysis of this robotic work cell, let $T^{\text{part}}_{\text{camera}}$ represent the position and orientation of the part as seen by the camera. For example:

$$T^{\text{part}}_{\text{camera}} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & -5 \\ 0 & 0 & -1 & 19 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3-8-1}$$

Next let $T^{\text{base}}_{\text{camera}}$ represent the position and orientation of the base of the robot as seen by the camera. In this case suppose that:

$$T^{\text{base}}_{\text{camera}} = \begin{bmatrix} 0 & -1 & 0 & 15 \\ -1 & 0 & 0 & 25 \\ 0 & 0 & -1 & 20 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3-8-2}$$

Consider the problem of finding the position and orientation of the part relative to the base of the robot. Using homogeneous coordinate transformation matrices, recalling that diagonal identifiers cancel, and using Prop. 2-4-1, we have:

$$\begin{aligned} T^{\text{part}}_{\text{base}} &= T^{\text{camera}}_{\text{base}} T^{\text{part}}_{\text{camera}} \\ &= (T^{\text{base}}_{\text{camera}})^{-1} T^{\text{part}}_{\text{camera}} \\ &= \begin{bmatrix} 0 & -1 & 0 & 15 \\ -1 & 0 & 0 & 25 \\ 0 & 0 & -1 & 20 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & -5 \\ 0 & 0 & -1 & 19 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -1 & 0 & 25 \\ -1 & 0 & 0 & 15 \\ 0 & 0 & -1 & 20 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & -5 \\ 0 & 0 & -1 & 19 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \left[\begin{array}{ccc:c} 1 & 0 & 0 & 30 \\ 0 & 1 & 0 & 15 \\ 0 & 0 & 1 & 1 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \end{aligned} \tag{3-8-3}$$

Thus the position of the part relative to the base frame of the robot is $p = [30, 15, 1]^T$. Furthermore, the three axes of the part frame point in the same directions as the corresponding axes of the base frame, as is indicated by the fact that the rotation submatrix is $R = I$. This is consistent with the picture in Fig. 3-12.

Next suppose we want to have the robot reach down and pick up the part directly from above by grasping it on the front and back faces. To determine the re-

quired rotation matrix $R = [x^t, y^t, z^t]$, first note from Fig. 3-12 that to pick up the part from above we need an approach vector which points down. In terms of base coordinates, this corresponds to $r^3 = -i^3$. Next, to grip the object on the front and back faces, we need a sliding vector of $r^2 = \pm i^2$. Finally, to complete the right-handed orthonormal coordinate frame, it is necessary that $r^1 = \mp i^1$. Thus there are two possible solutions for the rotation matrix R, one being:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{3-8-4}$$

We can now put the position and orientation information together to develop the tool configuration needed to grasp the part. In particular, the arm matrix which places the tool in the position p and orientation R to pick up the part is:

$$T_{\text{base}}^{\text{tool}}(q) = \left[\begin{array}{ccc:c} 1 & 0 & 0 & 30 \\ 0 & -1 & 0 & 15 \\ 0 & 0 & -1 & 1 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{3-8-5}$$

Alternatively, we can specify the more compact tool-configuration vector $w(q)$ needed to pick up the part. Here the required tool roll angle q_n must be determined. Suppose the robot in Fig. 3-12 is a Rhino XR-3. Then, from Eq. (3-4-2), we have $q_1 = \text{atan2}\ (15, 30) = 26.6$ degrees, and from Eqs. (3-8-4) and (3-4-20):

$$\begin{aligned} q_5 &= \text{atan2}\ (S_1 R_{11} - C_1 R_{21},\ S_1 R_{12} - C_1 R_{22}) \\ &= \text{atan2}\ (S_1, C_1) \\ &= 26.6 \text{ degrees} \end{aligned} \tag{3-8-6}$$

This is consistent with Fig. 3-7, where the robot is pictured in the soft home position, which corresponds to $q = [0, -\pi/2, \pi/2, 0, -\pi/2]^T$. From Def. 3-3-1, the tool-configuration vector $w(q)$ consists of the position vector p augmented by the approach vector r^3 scaled by $\exp(q_5/\pi)$. Thus, in this case, the tool configuration needed to grasp the block from above on the front and back faces is:

$$w = [30, 15, 1, 0, 0, -1.159]^T \tag{3-8-7}$$

To actually command the robot to reach down and pick up the part, we now need to know the value of the joint variables q that will produce the configuration of the tool specified in Eq. (3-8-7). At this point, Algorithm 3-4-1 would be used to determine the vector of joint variables q for the five-axis articulated robot.

Exercise 3-8-1: Focus Camera. Find the arm matrix needed to have the robot in Fig. 3-12 reach up and focus the camera by twisting the lens using tool roll motion.

3-9 PROBLEMS

3-1. Suppose a six-axis articulated robot has the following limits on the range of travel of its joint variables:

$$-3\pi/4 \le q_1 \le 3\pi/4 \qquad -\pi/2 \le q_4 \le \pi/2$$
$$-\pi/4 \le q_2 \le 3\pi/4 \qquad -\pi/2 \le q_2 + q_3 + q_5 \le \pi/2$$
$$-\pi/2 \le q_2 + q_3 \le \pi/2 \qquad -\pi \le q_6 \le \pi$$

Find the joint limit vectors $\{q^{\min}, q^{\max}\}$ and the joint coupling matrix C that characterize the joint-space work envelope Q of this robot.

3-2. Suppose the tool of a five-axis articulated robot is put into a *singular* configuration in which the tool roll axis is collinear with the base axis as shown in Fig. 3-13. Suppose q is a solution. Find Δq such that $q + \Delta q$ is also a solution. How many Δq are there? Is this a kinematically redundant robot?

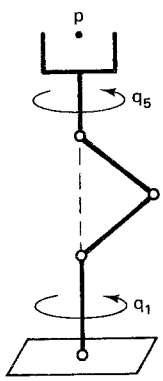


Figure 3-13 A singular tool configuration.

3-3. Consider the link-coordinate diagram for the four-axis cylindrical-coordinate robot with tool roll motion shown in Fig. 3-14. Here the vector of joint variables is $q = [\theta_1, d_2, d_3, \theta_4]^T$, and the kinematic parameters are as specified in Table 3-3. Find the arm matrix $T^{\text{tool}}_{\text{base}}(q)$ for this cylindrical-coordinate robot.

3-4. Find the tool-configuration vector $w(q)$ for the cylindrical-coordinate robot in Fig. 3-14.

3-5. Solve the inverse kinematics equations for the cylindrical-coordinate robot in Fig. 3-14. Do parts (a) through (d) in whichever order is easiest.

(a) Find the base angle $q_1(w)$.

(b) Find the vertical extension $q_2(w)$.

(c) Find the radial extension $q_3(w)$.

(d) Find the tool roll angle $q_4(w)$.

3-6. Find a reduced tool-configuration vector $\tilde{w}$ for the cylindrical-coordinate robot in Fig. 3-14.

3-7. For the cylindrical-coordinate robot in Fig. 3-14, use the rotation matrix $R(q)$ to find the tool roll angle q_4.

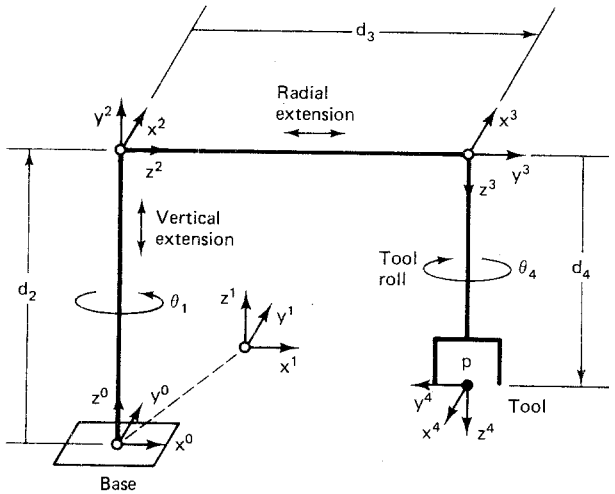


Figure 3-14 Link coordinates of a four-axis cylindrical robot.

TABLE 3-3 KINEMATIC PARAMETERS OF THE CYLINDRICAL ROBOT

Axis	θ	d	a	α	Home
1	q_1	0	0	0	0
2	$\pi/2$	q_2	0	$\pi/2$	$\bar{d}$
3	0	q_3	0	$\pi/2$	$\bar{r}$
4	q_4	d_4	0	0	π

3-8. Consider the link-coordinate diagram for the five-axis spherical-coordinate robot with tool pitch and tool roll motion shown in Fig. 3-15. Here the vector of joint variables is $q = [\theta_1, \theta_2, d_3, \theta_4, \theta_5]^T$, and the kinematic parameters are as specified in Table 3-4. Find the arm matrix $T_{\text{base}}^{\text{tool}}(q)$ for this spherical-coordinate robot.

3-9. Find the tool-configuration vector $w(q)$ for the spherical-coordinate robot in Fig. 3-15.

3-10. Solve the inverse kinematics equations for the spherical-coordinate robot in Fig. 3-15. Do parts (a) through (e) in whichever order is easiest.
- **(a)** Find the base angle $q_1(w)$.
- **(b)** Find the shoulder angle $q_2(w)$.
- **(c)** Find the radial extension $q_3(w)$.
- **(d)** Find the tool pitch angle $q_4(w)$.
- **(e)** Find the tool roll angle $q_5(w)$.

3-11. Find a reduced tool-configuration vector $\tilde{w}$ for the spherical-coordinate robot in Fig. 3-15.

3-12. For the spherical-coordinate robot in Fig. 3-15, use the rotation matrix $R(q)$ to find the tool roll angle q_5.

3-13. Recall from Chap. 2 that the composite yaw-pitch-roll transformation can be represented by the following rotation matrix R. Given values for the $\{R_{kj}\}$, solve for the yaw, pitch and roll angles $\theta \in \mathbf{R}^3$.

$$R(\theta) = \begin{bmatrix} C_2C_3 & S_1S_2C_3 - C_1S_3 & C_1S_2C_3 + S_1S_3 \\ C_2S_3 & S_1S_2S_3 + C_1C_3 & C_1S_2S_3 - S_1C_3 \\ -S_2 & S_1C_2 & C_1C_2 \end{bmatrix}$$

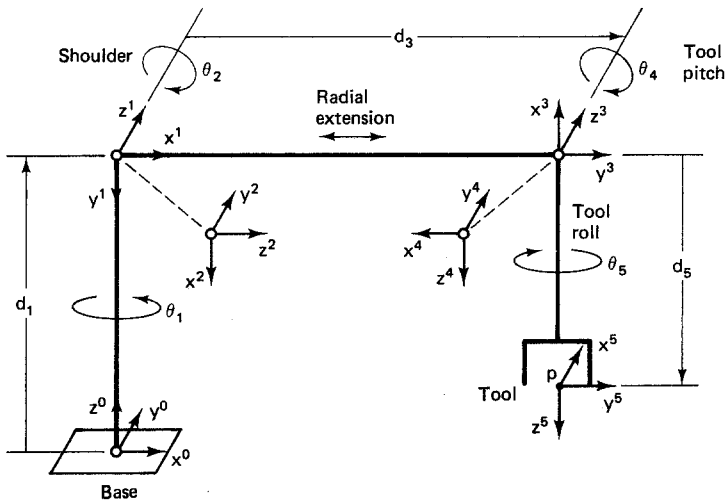


Figure 3-15 Link coordinates of a five-axis spherical robot.

TABLE 3-4 KINEMATIC PARAMETERS OF THE SPHERICAL ROBOT

Axis	θ	d	a	α	Home
1	q_1	d_1	0	$-\pi/2$	0
2	q_2	0	0	$\pi/2$	$\pi/2$
3	π	q_3	0	$\pi/2$	$\bar{r}$
4	q_4	0	0	$\pi/2$	$-\pi/2$
5	q_5	d_5	0	0	$\pi/2$

3-14. Suppose the joint-space work envelope Q for the four-axis cylindrical robot is of the following form. Show that there are two solutions to the inverse kinematics problem in this case by completing Table 3-5. The second solution should be expressed in terms of the first solution. For example, a joint variable might be negated, or π might be added to a joint variable.

$$-\pi \le q_1 \le \pi \qquad -H \le q_3 \le H$$
$$0 \le q_2 \le V \qquad -\pi \le q_4 \le \pi$$

TABLE 3-5 MULTIPLE SOLUTIONS OF THE FOUR-AXIS CYLINDRICAL ROBOT

Solution	Base	Vertical Extension	Radial Extension	Tool Roll
1	q_1	q_2	q_3	q_4
2				

3-15. Suppose the joint-space work envelope Q for the five-axis spherical robot is of the following form. Show that there are two solutions to the inverse kinematics problem in this case by completing Table 3-6. To simplify the problem somewhat, you can assume that $q_2 = 0$ corresponds to the arm pointing straight up.

$$-\pi \leq q_1 \leq \pi$$
$$-\pi \leq q_2 \leq \pi$$
$$0 \leq q_3 \leq H$$
$$-\pi \leq q_4 \leq \pi$$
$$-\pi \leq q_5 \leq \pi$$

TABLE 3-6 MULTIPLE SOLUTIONS OF THE FIVE-AXIS SPHERICAL ROBOT

Solution	Base	Shoulder	Radial Extension	Tool Pitch	Tool Roll
1	q_1	q_2	q_3	q_4	q_5
2					

REFERENCES

PIEPER, D. L. (1968). "The kinematics of manipulators under computer control," *AIM 72*, Stanford AI Laboratory, Stanford Univ.

UICKER, J. J., J. DENAVIT, and R. S. HARTENBERG (1964). "An iterative method for the displacement analysis of spatial mechanisms," *Trans. ASME J. Appl. Mech.*, Vol. 31, Ser. E, pp. 309–314.

WHITNEY, D. E. (1972). "The mathematics of coordinated control of prosthetic arms and manipulators," *J. Dynamic Systems, Measurement and Control*, Vol. 122, pp. 303–309.

4

Workspace Analysis and Trajectory Planning

Our investigation of robot arm kinematics has provided us with a mathematical model that can be used to place the tool, or end-effector, of the robot at an arbitrary position and orientation within the workspace. We now use this kinematics formulation as a steppingstone to a higher-level problem, the problem of planning trajectories for the tool to follow in order to perform meaningful manipulation tasks.

To properly formulate the trajectory planning problem, we begin with a more detailed examination of the robot workspace. The work envelopes of two generic classes of robots are examined. This is followed by a discussion of the important role played by workspace fixtures. Motion planning for the pick-and-place operation is then examined. This basic type of operation is associated with tasks such as machine loading and unloading. More sophisticated trajectory planning techniques are then discussed. These include continuous-path motion using inverse kinematics, interpolated motion using knot points, and straight-line motion. These more advanced methods are useful for applications such as spray painting, arc welding, and gluing.

4-1 WORKSPACE ANALYSIS

A rough analysis of work-envelope geometries was made in Chap. 1 based on the locus of points in $\mathbf{R}^3$ that could be reached by the robot wrist as the joint variables ranged over their respective limits. We now perform a more detailed analysis of the work envelope based on the positions reachable by the tool tip. In using the tool tip as a reference point, we must include the effects of both the major axes used to position the wrist and the minor axes used to orient the tool. When viewed as a subset of

$\mathbf{R}^3$, the shape or geometry of the work envelope varies from robot to robot. However, the work envelope can also be viewed within the framework of joint space $\mathbf{R}^n$. In joint space, the work envelope is typically characterized by bounds on linear combinations of joint variables. Constraints of this nature generate a convex polyhedron in $\mathbf{R}^n$. We call this polyhedron the *joint-space work envelope*.

Definition 4-1-1: Joint-Space Work Envelope. Let $q^{\min}$ and $q^{\max}$ be vectors in $\mathbf{R}^m$ denoting *joint limits* and let C be an $m \times n$ *joint coupling matrix*. Then the set of all values that the joint variables q can assume is called the *joint-space work envelope*. It is denoted Q and is of the form:

$$Q \triangleq \{q \in \mathbf{R}^n: q^{\min} \leq Cq \leq q^{\max}\}$$

The simplest special case of a joint-space work envelope occurs when $C = I$, which represents no interaxis coupling. In this case the number of constraints, m, matches the number of joint variables, n, with $q_k^{\min} \leq q_k \leq q_k^{\max}$ for $1 \leq k \leq n$. In general, the joint coupling matrix C can be rectangular ($m \neq n$) and can contain off-diagonal terms. This occurs, for example, when there are bounds on linear combinations (sums or differences) of joint variables.

Exercise 4-1-1: Joint-Space Work Envelope. A subset U of $\mathbf{R}^n$ is *convex* if and only if for any two points x^1 and x^2 in U, the line segment connecting x^1 to x^2 is also in U, that is, $x(\lambda) = (1 - \lambda)x^1 + \lambda x^2$ is in U for $0 \leq \lambda \leq 1$. Show that the joint-space work envelope Q is a convex subset of joint space.

Although the joint-space work envelope Q is relatively easy to characterize, it is the locus of points in $\mathbf{R}^3$ that can be reached by the tool tip that is of primary interest. There are a number of different types of work envelopes in $\mathbf{R}^3$ that might be considered at this point. For example, the locus of all points reachable from *at least one* tool orientation is referred to as the *total* work envelope, or simply the work envelope, and is denoted Υ. Alternatively, the locus of points reachable from an *arbitrary* tool orientation is called the *dexterous* work envelope and is denoted Υ_d (Beni and Hackwood, 1985). Clearly, the dexterous work envelope is smaller, but more practical, than the total work envelope. The work envelope of a manipulator can be characterized directly in terms of the joint-space work envelope Q and the tool tip position function $p(q)$ as follows:

Proposition 4-1-1: Work Envelope. Let Q denote the joint-space work envelope of a robotic manipulator. The *work envelope* Υ is the locus of all points $p \in \mathbf{R}^3$ that are reachable by the tool tip. It can be expressed in terms of Q as:

$$\Upsilon = \{p(q) \in \mathbf{R}^3: q \in Q\}$$

We characterize the work envelope Υ mathematically by saying that it is the *image* of the joint-space work envelope Q under the tool position function $p(q)$. Recall that the tool position function corresponds to the fourth column of the arm matrix $T_{\text{base}}^{\text{tool}}(q)$. Alternatively, the tool position function consists of the first three

components of the tool-configuration vector $w(q)$. Recall from Def. 3-3-1 that the tool-configuration vector is $w = (w^1, w^2)$ where:

$$w^1(q) = p(q) \tag{4-1-1}$$

$$w^2(q) = [\exp(q_n/\pi)]r^3(q) \tag{4-1-2}$$

In the approach to workspace analysis presented here, we use the tool orientation angles implicit in w^2 as *parameters* of the work envelope. Thus we generate a *family* of work envelopes $\{Y(w^2)\}$ with each member of the family associated with a particular tool orientation. The motivation for this approach lies in the observation that we typically have some sense of how we want to approach a part in order to grasp or manipulate it. Once the tool orientation is selected, we can then determine the range of locations in the workspace over which such a manipulation is possible. There is no single closed-form solution to the problem of determining the work envelope that is applicable to robotic manipulators in general. Since the detailed solution of the work envelope Y is robot-dependent, we focus our attention for illustration purposes on a number of generic classes of robots.

4-2 WORK ENVELOPE OF A FIVE-AXIS ARTICULATED ROBOT (RHINO XR-3)

The first type of robot we examine is a five-axis articulated robot with tool pitch and tool roll motion. This class of manipulators includes, for example, the Rhino XR-3 robot and the Microbot Alpha II robot as special cases. The link-coordinate diagram for a five-axis articulated robot, previously developed in Chap. 2 and shown as Fig. 2-23, is repeated for convenient reference in Fig. 4-1.

To investigate the work envelope of this manipulator, we start with the expression for the tool-configuration vector $w(q)$. Recall that for an articulated robot the

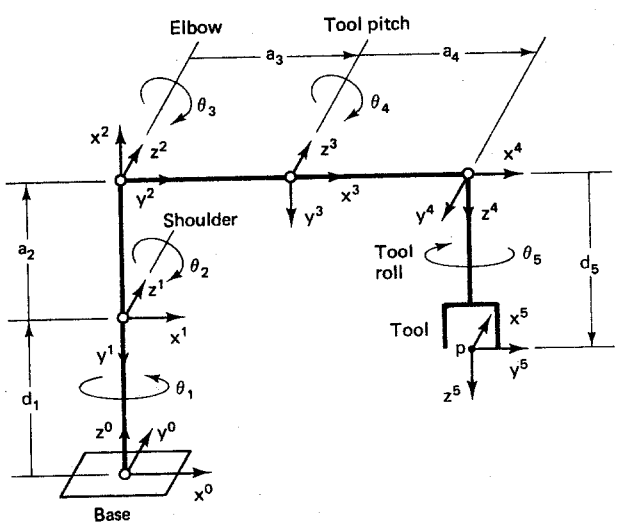


Figure 4-1 Link coordinates of a five-axis articulated robot (Rhino XR-3).

vector of joint variables is simply $q = \theta$. From the analysis performed in Chap. 3, the expression for the tool-configuration vector of the five-axis articulated robot in Fig. 4-1 is:

$$w(q) = \begin{bmatrix} C_1(a_2C_2 + a_3C_{23} + a_4C_{234} - d_5S_{234}) \\ S_1(a_2C_2 + a_3C_{23} + a_4C_{234} - d_5S_{234}) \\ d_1 - a_2S_2 - a_3S_{23} - a_4S_{234} - d_5C_{234} \\ \hline -[\exp(q_5/\pi)]C_1S_{234} \\ -[\exp(q_5/\pi)]S_1S_{234} \\ -[\exp(q_5/\pi)]C_{234} \end{bmatrix} \tag{4-2-1}$$

The joint variables of the minor axes used to orient the tool are the tool pitch angle q_4 and the tool roll angle q_5. However, it is evident from inspection of Eq. (4-2-1) that the tool roll angle has no effect on the tool-tip position p. Consequently, the tool roll angle can be ignored in our investigation of the size and shape of the work envelope. However, the tool-tip position does change with the pitch angle q_4. In planning a manipulation, usually the orientation of the part to be manipulated is known relative to the base frame of the robot. Consequently, it is the tool pitch angle measured relative to the x^0y^0 plane that is the germane measure of orientation in this case. This is the *global tool pitch* angle q_{234}. We will assume that the global tool pitch angle has been determined on the basis of an analysis of the part orientation. Furthermore, for the purposes of illustration, we assume that the joint angles are constrained to lie in the following joint-space work envelope:

$$\begin{bmatrix} -\pi \\ -3\pi/4 \\ \pi/4 \\ -5\pi/4 \\ -\pi \\ -3\pi/4 \\ -5\pi/4 \end{bmatrix} \leq \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} q \leq \begin{bmatrix} \pi \\ \pi/4 \\ 3\pi/4 \\ \pi/4 \\ \pi \\ \pi/4 \\ \pi/4 \end{bmatrix} \tag{4-2-2}$$

Note that there is effectively no constraint on the base angle q_1; it is allowed to range over one complete cycle $[-\pi, \pi]$. The constraints on the shoulder angle q_2, elbow angle q_3, and tool pitch angle q_4 prevent adjacent links from colliding. The tool roll angle q_5, like the base angle, ranges over one complete cycle $[-\pi, \pi]$. The constraints on the global elbow angle q_{23} limit the orientation of the forearm relative to the work surface. For example, $q_{23} = 0$ corresponds to the forearm horizontal, whereas $q_{23} = -\pi/2$ corresponds to the forearm vertical and pointing up. Similarly, the constraints on the global tool pitch angle q_{234} limit the orientation of the tool relative to the work surface. In this case, $q_{234} = 0$ corresponds to the approach vector pointing straight down, and $q_{234} = -\pi/2$ corresponds to it pointing straight out.

Exercise 4-2-1: Maximum Reach. Show that the joint-space work envelope

Q represented by Eq. (4-2-2) includes arm configurations corresponding to the maximum horizontal reach and the maximum vertical reach. In each case, for what values of q is the arm fully extended?

Once the global tool pitch angle q_{234} has been selected, a simple equation which specifies the *outer surface* of the work envelope Y can then be determined. The basic constraint on the reach of an articulated robot is illustrated in Fig. 4-2. Here the distance from the shoulder frame L_2 to the wrist frame L_4 can never be larger than $a_2 + a_3$, where a_2 is the length of the upper arm and a_3 is the length of the forearm. This can be expressed algebraically as:

$$b_1^2 + b_2^2 \leq (a_2 + a_3)^2 \tag{4-2-3}$$

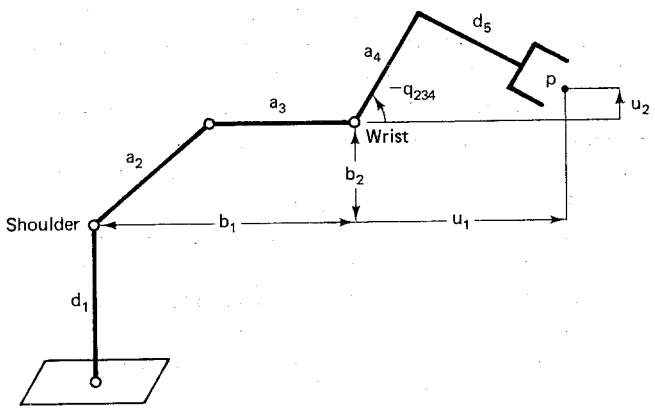


Figure 4-2 Evaluating the reach of a five-axis articulated robot.

Here b_1 and b_2 represent the horizontal and vertical components, respectively, of the distance between the shoulder frame and the wrist frame as illustrated in Fig. 4-2.

The horizontal and vertical components b_1 and b_2 can be expressed in terms of the tool-tip position p and the global tool pitch angle q_{234}. To see this, we introduce two intermediate variables:

$$u_1 \triangleq a_4 C_{234} - d_5 S_{234} \tag{4-2-4a}$$

$$u_2 \triangleq -a_4 S_{234} - d_5 C_{234} \tag{4-2-4b}$$

Here u_1 and u_2 are known constants once the tool orientation q_{234} has been selected. From Fig. 4-2 we see that the constant u_1 represents the horizontal displacement between the tool-tip position p and the origin of the wrist frame. Similarly, the constant u_2 represents the vertical displacement between the tool-tip position p and the origin of the wrist frame. The basic constraint which defines the *outer surface* of the work envelope, or *reach* of the robot, can now be expressed by rewriting Eq. (4-2-3) using the orientation constants in Eq. (4-2-4).

Proposition 4-2-1: Reach Constraint. For the articulated robot shown in

Fig. 4-1, once the global tool pitch angle q_{234} has been selected, the tool-tip position p must satisfy the following reach constraint, where u is as in Eq. (4-2-4):

$$[(p_1^2 + p_2^2)^{1/2} - u_1]^2 + (p_3 - d_1 - u_2)^2 \leq (a_2 + a_3)^2$$

Recall that the constants u_1 and u_2 are orientation-dependent, since they change with q_{234}. Thus Prop. 4-2-1 is an outer-surface constraint for a family of work envelopes $\{Y(q_{234})\}$ parameterized by the global tool pitch angle q_{234}. At this point we can now place limits on the components of the position vector p. The individual limits depend upon the order in which the components of p are selected. We will assume, for the purposes of illustration, that they are selected in the following order:

Limits on p_1. Inspection of the reach constraint in Prop. 4-2-1 reveals that the maximum value of $|p_1|$ occurs when $(p_2, p_3) = (0, d_1 + u_2)$. Thus the upper bound on $|p_1|$, for a given tool orientation q_{234}, reduces to:

$$|p_1| \leq a_2 + a_3 + u_1 \tag{4-2-5}$$

Limits on p_2. Once the first coordinate of p has been selected, we can make use of the reach constraint to establish an upper bound on $|p_2|$. From Prop. 4-2-1, it is evident that the maximum value of $|p_2|$ occurs when $p_3 = d_1 + u_2$. Thus, given a tool orientation q_{234} and a value for p_1, we then have the following upper bound on $|p_2|$:

$$|p_2|^2 \leq (a_2 + a_3 + u_1)^2 - p_1^2 \tag{4-2-6}$$

Note from Eq. (4-2-5) that if p_1 is selected to be its maximum possible value, then the constraint on p_2 reduces to $p_2 = 0$. That is, in selecting the maximum possible value for p_1, we remove all of the freedom that would otherwise be available for choosing p_2.

Limits on p_3. We can put a bound on p_3 by again making use of the reach constraint. In this case the tool orientation q_{234} and both p_1 and p_2 have already been selected. Thus, from Prop. 4-2-1 we have:

$$(p_3 - d_1 - u_2)^2 \leq (a_2 + a_3)^2 - [(p_1^2 + p_2^2)^{1/2} - u_1]^2 \tag{4-2-7}$$

Again, note from Eqs. (4-2-5) and (4-2-6) that if either p_1 or p_2 is selected to be its maximum possible value, then the constraint on p_3 reduces to $p_3 = d_1 + u_2$. Thus, in selecting the maximum possible value for p_1 or p_2, we remove all of the freedom that would otherwise be available for choosing p_3.

The upper bounds on the tool-tip position p specified in Eqs. (4-2-5) to (4-2-7) represent the outer surface of the work envelope. Clearly, there is also an *inner surface* that must not be penetrated. For example, if $(p_1, p_2) = (0, 0)$, then the tool tip is touching the base axis of the robot. Similarly, if $p_3 < 0$, then the tool tip is below the x^0y^0 plane, or work surface! To place lower bounds on the components of p we must know more about the size and shape of the tool. For example, we must know the maximum tool radius and the total tool length, including any portion of the tool that extends behind the pitch axis.

Exercise 4-2-2: Work Surface. Suppose the tool of the Rhino XR-3 robot can be enclosed in a cylinder of radius 75 mm and length 300 mm concentric about the tool roll axis and extending 100 mm behind the tool pitch axis. Assuming that the global tool pitch q_{234} is restricted to the range $[-\pi/2, 0]$, find bounds on p_3 which ensure that the cylinder enclosing the tool will not penetrate the work surface, or x^0y^0 plane.

Exercise 4-2-3: Inside Surface. Suppose the body of the Rhino XR-3 robot can be enclosed in a cylinder of radius 125 mm concentric about the base axis. Given the cylinder enclosing the tool and the constraints on q_{234} specified in Exercise 4-2-2, find bounds on p_1 and p_2 that ensure that the tool cylinder will not penetrate the robot body cylinder.

4-3 WORK ENVELOPE OF A FOUR-AXIS SCARA ROBOT (ADEPT ONE)

Another important class of robotic manipulators is the four-axis horizontal-jointed robot, or SCARA robot. The link-coordinate diagram for the four-axis SCARA robot, developed in Chap. 2 and shown in Fig. 2-25, is displayed in Fig. 4-3 for convenient reference. In this case, the vector of joint variables is $q = [\theta_1, \theta_2, d_3, \theta_4]^T$, where the third joint is prismatic while the remaining three joints are revolute. From the analysis performed in Chap. 3, the expression for the tool-configuration vector of the four-axis SCARA robot in Fig. 4-3 is:

$$w(q) = \begin{bmatrix} a_1C_1 + a_2C_{1-2} \\ a_1S_1 + a_2S_{1-2} \\ d_1 - q_3 - d_4 \\ \hline 0 \\ 0 \\ -\exp(q_4/\pi) \end{bmatrix} \tag{4-3-1}$$

The one joint variable used to orient the tool is the tool roll angle q_4. However, it is evident from inspection of Eq. (4-3-1) that the tool roll angle has no effect on the tool-tip position p, which means that q_4 can be ignored in our investigation of the work envelope. It follows that the family of work envelopes in this case has only one member. For the purposes of illustration, we assume that the joint variables of the SCARA robot are constrained to lie in the following joint-space work envelope, where $\beta \geq 0$:

$$\begin{bmatrix} -\pi \\ -\pi + \beta \\ h \\ -\pi \end{bmatrix} \leq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} q \leq \begin{bmatrix} \pi \\ \pi - \beta \\ H \\ \pi \end{bmatrix} \tag{4-3-2}$$

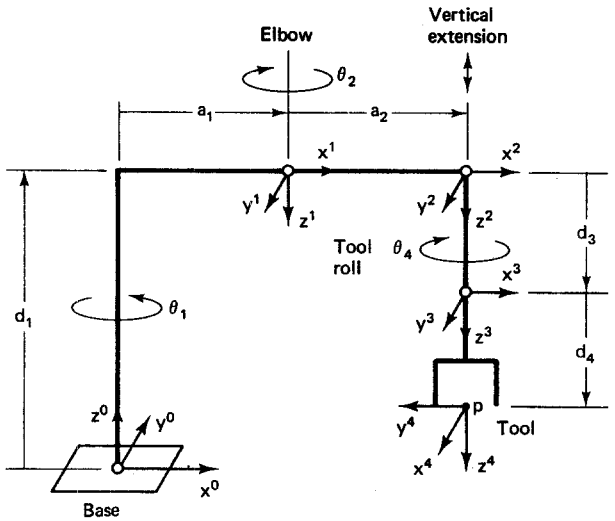


Figure 4-3 Link coordinates of a four-axis SCARA robot (Adept One).

Note that there is no effective restraint on the base angle q_1; it varies over the complete range $[-\pi, \pi]$. The limits on the elbow angle of $|q_2| \leq \pi - \beta$ prevent the upper arm and forearm from colliding if $\beta > 0$, assuming that these links are in the same horizontal plane ($d_2 = 0$). The limit on q_3 to the interval $[h, H]$ effectively specifies the vertical stroke of the manipulator. There is no restraint on the tool roll angle q_4; it, too, varies over the complete range $[-\pi, \pi]$.

Simple constraints which specify the boundaries of the work envelope can now be specified in terms of the joint limits. Robots are usually designed in such a way that the links become smaller and lighter as one moves toward the distal end of the arm. This tends to reduce the size of the actuators of the proximal joints near the base. For the purposes of illustration, we will therefore assume that the upper arm a_1 is at least as long as the forearm a_2. If the base angle q_1 is unrestricted, as in Eq. (4-3-2), then the maximum and minimum horizontal reach of the SCARA robot are as illustrated in Fig. 4-4.

The maximum horizontal reach of the SCARA robot, r_{max}, is clearly $a_1 + a_2$. The minimum horizontal reach r_{min} can be computed using the trigonometric identi-

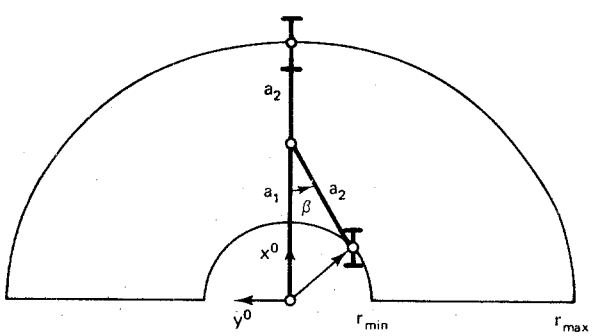


Figure 4-4 Maximum and minimum horizontal reach of a SCARA robot.

ties in Appendix 1. From Fig. 4-4, we have $r_{\min}^2 = a_1^2 + a_2^2 - 2a_1a_2 \cos \beta$. Given the constraints on the prismatic joint in Eq. (4-3-2), we see from Fig. 4-3 that the maximum vertical reach is $d_1 - d_4 - h$, while the minimum vertical reach is $d_1 - d_4 - H$. We can therefore summarize the work envelope of a SCARA robot as follows:

Proposition 4-3-1: SCARA Work Envelope. For the SCARA robot shown in Fig. 4-3, if $a_1 \geq a_2$ and q satisfies Eq. (4-3-2), then the locus of points p that are reachable by the tool tip satisfies the following inequalities:

$$a_1^2 + a_2^2 - 2a_1a_2 \cos \beta \leq p_1^2 + p_2^2 \leq (a_1 + a_2)^2$$

$$d_1 - d_4 - H \leq p_3 \leq d_1 - d_4 - h$$

Note that when the base joint is allowed to travel over its full range $[-\pi, \pi]$, the work envelope is the volume between two concentric cylinders centered at the base axis. Thus a SCARA robot is similar to a cylindrical robot in this case. The height of the cylinder is the vertical stroke of the manipulator, namely, $H - h$. The outer radius $r_{\max}$ is the length of the upper arm plus the length of the forearm. Finally, the inner radius $r_{\min}$ depends on the relative lengths of the upper arm and forearm and the range of travel of the elbow. When the range of the base angle q_1 is less than $[-\pi, \pi]$, the horizontal cross section of the work envelope becomes more complex. For example, a horizontal cross section of the work envelope Y when q_1 is restricted to $[-\pi/2, \pi/2]$ and q_2 is unrestricted ($\beta = 0$) is shown in Fig. 4-5.

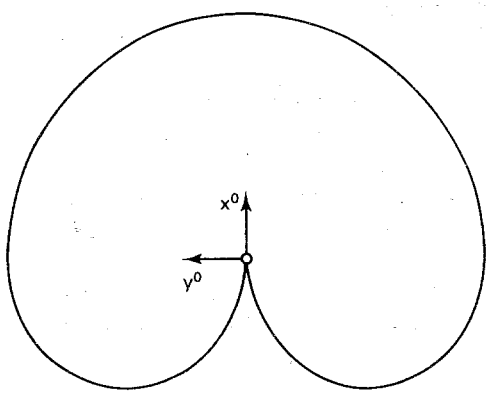


Figure 4-5 Horizontal cross section of restricted SCARA work envelope.

Exercise 4-3-1: Long Forearm. Suppose q satisfies Eq. (4-3-2) and $a_2 > a_1$. Find the horizontal reach and stroke of the SCARA robot in this case.

4-4 WORKSPACE FIXTURES

Automated manufacturing lines typically include a number of robotic work cells arranged to optimize the flow of materials and products. If the robotic manipulators are to be used effectively, careful consideration must be given to the design and layout of workspace fixtures for each robotic work cell. Workspace fixtures take the

form of *part feeders* used to present parts to the robot, *transport devices* used to move parts from one robotic work cell to another, and *part-holding devices* used in robotic assembly tasks. Part fixturing is a significant issue if for no other reason than the observation that, in many instances, the costs associated with designing and developing custom part-fixturing devices can easily exceed the purchase price of the robot itself!

4-4-1 Part Feeders

If a robot is to be successfully used to manipulate a part, the part must be presented to the robot in a manner that is compatible with the tool configurations realizable by the manipulator. Often several different types of parts must be presented to a robot, say, for a subassembly task. In cases like these, the part feeders might be arranged *concentrically* around the robot base near the outer boundary of the work envelope, as shown in Fig. 4-6. This way the assembly work can be done in the space closer to the robot, where the robot precision, with respect to the base axis, is highest.

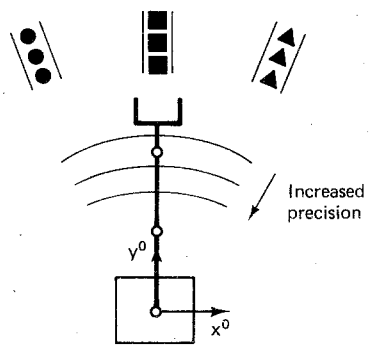


Figure 4-6 Concentric layout of multiple part feeders.

Some part feeders are activated electromechanically, while others are activated by gravity. Electrically activated part feeders include *vibratory-bowl* devices, in which the parts, through agitation, work their way through a chute or tube until they come out the end where they are presented to the robot. Vibratory-bowl part feeders have the advantage that the supply of parts can often be simply dumped into the top of the bowl. However, when the parts come out the chute, they are not necessarily presented with a uniform orientation.

A simple, common type of gravity-fed part feeder consists of an inclined tube or slide on which the parts are placed end to end in identical orientations, as shown in Fig. 4-7. The *incline angle* β is made steep enough that whenever a part is removed from the bottom, the remaining parts immediately slide down to fill the vacated space. In particular, if μ is the coefficient of friction between the part and the incline, then:

$$\beta > \text{atan2}\ (\mu,\ 1) \tag{4-4-1}$$

As an illustration of a kinematic analysis of a part feeder of this type, consider the problem of configuring the tool to remove a part from the bottom of the part feeder shown in Fig. 4-7. We begin by determining the rotation matrix R needed to

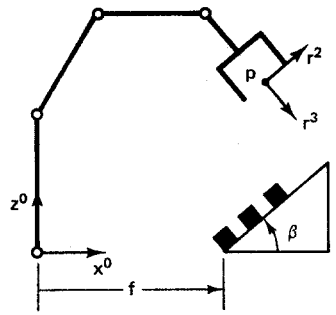


Figure 4-7 A gravity-fed part feeder.

properly orient the tool. Typically, a part is approached along a line that is *orthogonal* to the work surface on which the part rests. If we assume that the part feeder is aligned with the x^0 axis, then the approach vector r^3 will be in the x^0z^0 plane. The x and z components of r^3 can be computed using the diagram shown in Fig. 4-8.

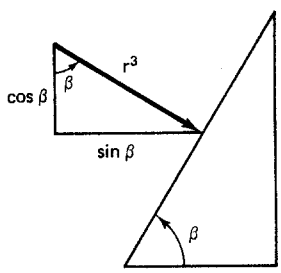


Figure 4-8 Computing the approach to an inclined slide.

Using similar triangles, we find that $r^3 = [S\beta, 0, -C\beta]^T$, where $S\beta = \sin\beta$ and $C\beta = \cos\beta$. Next, to establish the sliding vector, recall that the parts are stacked end to end on the inclined surface with no space between them. Consequently, the part at the bottom should be grasped along its sides. From Fig. 4-7, this dictates that the sliding vector should be $r^2 = \pm i^2$. Suppose we choose $r^2 = -i^2$. The normal vector $r^1 = r^2 \times r^3$ must be orthogonal to both r^2 and r^3 in a right-handed sense. Thus the proper orientation for the tool to pick a part off the bottom of the feeder in Fig. 4-7 is:

$$R = \begin{bmatrix} C\beta & 0 & S\beta \\ 0 & -1 & 0 \\ S\beta & 0 & -C\beta \end{bmatrix} \tag{4-4-2}$$

Next consider the tool-tip position vector p needed to properly grasp the part. Suppose the bottom of the part feeder is located f units along the x^0 axis, as shown in Fig. 4-9. Furthermore, suppose the part being grasped is a cube of dimension c. If the cube is to be grasped at its center, then, using similar triangles, the coordinates of the centroid of the cube are found to be $p = [f + c(C\beta - S\beta)/2, 0, c(S\beta + C\beta)/2]^T$. Combining the centroid calculation with Eq. (4-4-2), this results in the following value for the arm matrix at the pick position:

$$T_{\text{base}}^{\text{pick}} = \left[\begin{array}{ccc|c} C\beta & 0 & S\beta & f + c(C\beta - S\beta)/2 \\ 0 & -1 & 0 & 0 \\ S\beta & 0 & -C\beta & c(C\beta + S\beta)/2 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{4-4-3}$$

Recall from Chap. 3 that if we now want to command the robot to reach down to position itself to grasp the part, we would use $T_{\text{base}}^{\text{pick}}$ as the input to the inverse kinematics procedure in Algorithm 3-4-1. This would then yield the joint variable vector q needed to properly configure the tool to remove a block from the bottom of the feeder.

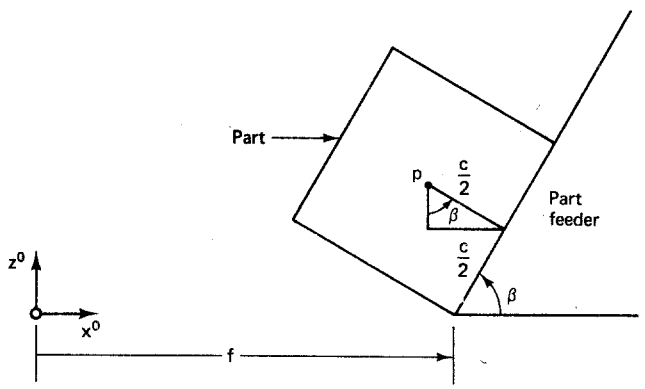


Figure 4-9 Computing the position to grasp a cube.

Exercise 4-4-1: Feeder Location 1. Use the reach constraint in Prop. 4-2-1 to find an upper bound on f that ensures that the part on the bottom of the feeder in Fig. 4-7 is reachable by the robot at the proper tool orientation.

Exercise 4-4-2: Part Insertion. Find the arm matrix value $T_{\text{base}}^{\text{place}}$ needed to place a part at the top of the feeder in Fig. 4-7, assuming that the feeder has a capacity of n cubes of dimension c.

Exercise 4-4-3: Feeder Location 2. Use the reach constraint in Prop. 4-2-1 to find an upper bound on f that ensures that a part can be removed from the bottom of the feeder and placed on the top, assuming that the feeder has a capacity of n cubes of dimension c.

4-4-2 Conveyors and Carousels

Conveyors and carousels can also be regarded as part feeders. However, they are special part feeders, in the sense that they do more than simply present parts to a robot for manipulation. They also *transport* parts between robots and from one robotic work cell to another. Conveyor belts are *linear* transport devices, whereas carousels are *rotary* transport devices.

An illustration of a conveyor system used in a multirobot work cell is shown in Fig. 4-10. Here the conveyor belt transports the parts horizontally from left to right,

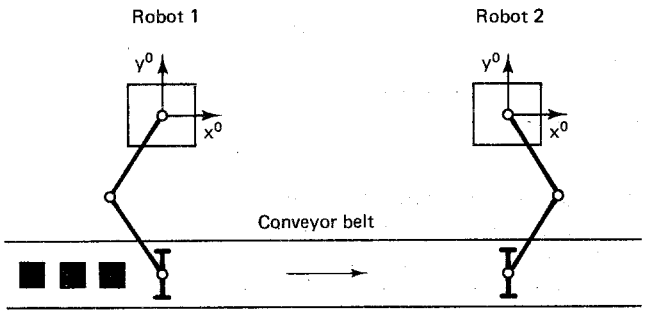


Figure 4-10 A linear transport device (conveyor).

and the parts are approached from above with $r^3 = -i^3$. Although a five- or six-axis robot could be used in this case, the extra axes are really not necessary. Whenever parts are manipulated from directly above, a four-axis horizontal-jointed, or SCARA-type, robot is sufficient. The first two axes position the tool over the conveyor belt, the third axis places the tool at the proper elevation, and the fourth axis rolls the tool to establish the proper sliding vector.

As an illustration of how a rotary transport device might be used to transfer parts back and forth between two robots, consider the carousel shown in Fig. 4-11. Here the parts are again approached directly from above, and in this case, therefore, four-axis SCARA robots can be used.

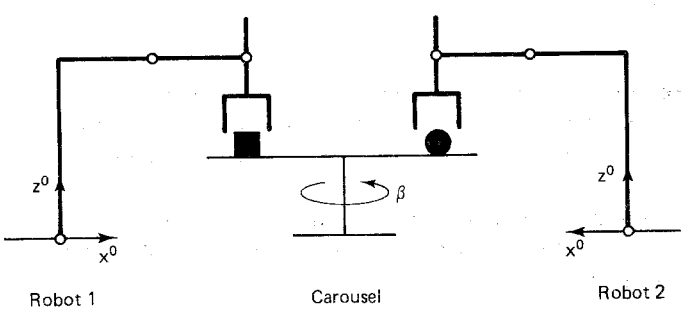


Figure 4-11 A rotary transport device (carousel).

Note that parts can be easily exchanged between the two robots even when the carousel can turn in only one direction. This is not the case with a linear transport device such as a conveyor, where the belt direction has to be reversed if an exchange or swap of parts is to take place. However, one important advantage that a conveyor system does have is that it can be easily designed to accommodate *many* robots. A rotary transport device such as a carousel is limited to a relatively few robots, because otherwise the diameter of the turntable would become too large.

4-4-3 Fixed Tools

A growing number of robotic applications involve some sort of assembly task. Since most robots are one-handed, and therefore have only one tool in operation at any given time, special work-holding fixtures are needed to secure the subassembly at a

workstation. The robot then manipulates the new part being added to the subassembly. Work-holding fixtures typically consist of specialized jigs built to hold a particular type of part or subassembly in place at the proper position and orientation (Boyes, 1985).

A work-holding fixture can actually be regarded as a *fixed tool.* It is fixed or immobilized in the sense that it cannot move or reorient itself, except perhaps to make fine adjustments. However, it can grasp and release parts presented to it. As an illustration of how a fixed tool might be used for a simple assembly task, consider the problem of threading a hex nut on a bolt. Suppose the robot has already picked up the bolt and placed it in a fixed tool, or computer-controlled clamp, as shown in Fig. 4-12.

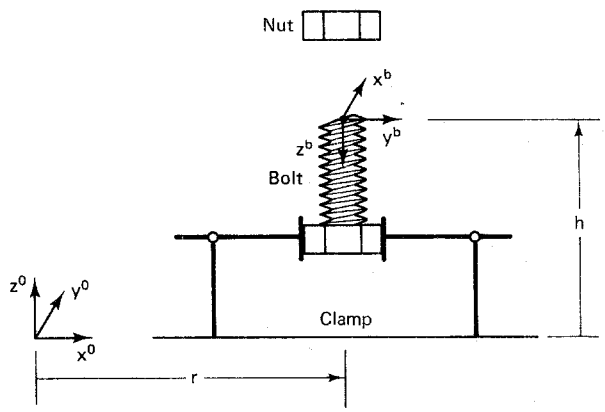


Figure 4-12 A fixed tool for part holding.

In planning a nut-fastening trajectory for the tool, we begin with the assumption that the nut will be twisted using the tool roll motion of the robot. This dictates that the approach vector must be constant and pointing down, or that $r^3 = -i^3$. The normal and sliding vectors must rotate in a clockwise sense from above if the bolt has a right-handed thread. To determine the speed of the threading operation, suppose it takes T seconds to roll the tool one complete revolution. If a screw transformation is applied to the bolt frame $B = \{x^b, y^b, z^b\}$ shown in Fig. 4-12, then the required angular displacement of the screw is:

$$\phi(t) = \frac{2\pi t}{T} \tag{4-4-4}$$

To determine the linear displacement of the screw transformation, we must specify how fast the tool has to slide down bolt axis z^b as the nut is turned. Suppose the *pitch* of the bolt thread is ρ threads per millimeter. Since the tool turns one rotation every T seconds, the tool must move down the bolt axis with a linear displacement of

$$\lambda(t) = \frac{t}{\rho T} \quad \text{mm} \tag{4-4-5}$$

From Fig. 4-12, the thread is b millimeters long. Setting $\lambda(t) = b$ and solving for t yields a total threading time of $t = b\rho T$. Thus the tool must execute the following screw transformation with respect to the bolt frame B to tighten the nut:

$$T_{\text{bolt}}^{\text{thread}}(t) = \text{Screw}\,[\lambda(t),\ \phi(t),\ 3] \qquad 0 \le t \le b\rho T. \tag{4-4-6}$$

Next we must determine the tool trajectory relative to the base frame, $\{x^0, y^0, z^0\}$. From Fig. 4-12, we see that the bolt frame origin is at coordinates $p = [r,\ 0,\ h]^T$ with respect to the base frame. Bolt axis x^b is aligned with base axis y^0, bolt axis y^b is aligned with base axis x^0, and bolt axis z^b points in the opposite direction from base axis z^0. Thus the homogeneous transformation matrix which maps bolt coordinates into base coordinates is:

$$T_{\text{base}}^{\text{bolt}} = \left[\begin{array}{ccc:c} 0 & 1 & 0 & r \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & h \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{4-4-7}$$

The nut-threading operation is formulated by combining Eqs. (4-4-4) through (4-4-7). Recalling the definition of the kth fundamental homogenous screw transformation (Def. 2-4-2), we then have for $0 \le t \le b\rho T$:

$$\begin{aligned} T_{\text{base}}^{\text{thread}}(t) &= T_{\text{base}}^{\text{bolt}}\, T_{\text{bolt}}^{\text{thread}}(t) \\ &= T_{\text{base}}^{\text{bolt}}\ \text{Screw}\,[\lambda(t),\ \phi(t),\ 3] \\ &= T_{\text{base}}^{\text{bolt}}\ \text{Rot}\,[\phi(t),\ 3]\ \text{Tran}\,[\lambda(t) i^3] \\ &= \begin{bmatrix} 0 & 1 & 0 & r \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos[\phi(t)] & -\sin[\phi(t)] & 0 & 0 \\ \sin[\phi(t)] & \cos[\phi(t)] & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \lambda(t) \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \left[\begin{array}{ccc:c} \sin(2\pi t/T) & \cos(2\pi t/T) & 0 & r \\ \cos(2\pi t/T) & -\sin(2\pi t/T) & 0 & 0 \\ 0 & 0 & -1 & h - t/\rho T \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \end{aligned} \tag{4-4-8}$$

Note that this is an instance where straight-line motion of the tool is required. In view of the form of the rotation matrix, this threading operation can again be carried out with a SCARA robot. If the tool roll is limited to, say, $[-\pi, \pi]$, then the nut must be fastened one turn at a time.

Exercise 4-4-4: Nut Removal. Find $T_{\text{base}}^{\text{unthread}}(t)$, the trajectory needed to implement an unthreading operation in Fig. 4-12. If the threaded portion of the bolt is b millimeters long, how long will it take for the nut to be removed, assuming $1/T$ turns per second?

4-5 THE PICK-AND-PLACE OPERATION

Perhaps the most fundamental robotic manipulation task is the *pick-and-place operation*. This is the type of motion that is needed, for example, in the automated loading and unloading of machines. More generally, pick-and-place motions are used to alter the distribution of parts within the workspace. Simple point-to-point motion control can usually be used to execute pick-and-place operations, with the *minimal* pick-and-place trajectory consisting of four discrete points, as follows.

4-5-1 Pick and Lift-Off Points

At one end of the pick-and-place trajectory is the *pick point,* whose coordinate frame is denoted $T_{\text{base}}^{\text{pick}}$. This represents the initial position and orientation of the part being manipulated. The pick position, p^{pick}, is often taken to be the center of mass, or *centroid,* of the part. Alternatively, if the part is of an irregular shape, then the centroid of some *feature* of the part such as a handle might be used. The pick orientation, R^{pick}, must also be specified. Usually, the tool rotation matrix is selected to ensure that the approach vector is *orthogonal* to the work surface on which the part is resting. When possible. the sliding vector should be selected so as to grasp the part along two *parallel* faces where the separation between the faces is smaller than the maximum opening of the tool. A typical tool configuration for the pick point is shown in Fig. 4-13, where the work surface is horizontal.

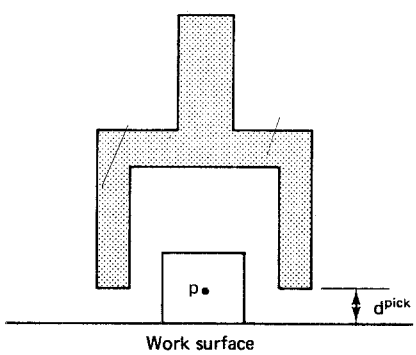


Figure 4-13 Tool configuration at the pick point.

The second point of a pick-and-place trajectory is the *lift-off point,* whose coordinate frame is denoted $T_{\text{base}}^{\text{lift}}$. The lift-off point is a point *near* the pick point that the robot moves to, initially, before an attempt is made to reach down and pick up the part. In this way a *gross motion* to the general vicinity of the part can be made first. This is followed by a *fine motion,* in which the part is carefully approached with the tool in the proper orientation. Whereas the pick point is specified by the user, or is perhaps computed with a vision system, the lift-off point is an intermediate point that is *inferred* from the pick point. More specifically, once the pick point frame $T_{\text{base}}^{\text{pick}}$ is determined, the associated lift-off point frame can then be computed as follows:

$$T_{\text{base}}^{\text{lift}} = \left[\begin{array}{c|c} R^{\text{pick}} & p^{\text{pick}} - vR^{\text{pick}}i^3 \\ \hline 0\ \ 0\ \ 0 & 1 \end{array}\right] \tag{4-5-1}$$

Note that the tool orientation at the lift-off point is identical to the tool orientation at the pick point; that is, $R^{\text{lift}} = R^{\text{pick}}$. This way the tool orientation *remains fixed* as the tool moves from the lift-off point to the pick point. The tool position at the lift-off point is obtained by starting at the pick position and moving backward a distance v along the approach vector. Thus the lift-off point will be a safe distance, v, away from the pick point. We refer to v as the *approach distance*.

4-5-2 Place and Set-Down Points

A third point of the pick-and-place trajectory is the *place point*, whose coordinate frame is denoted $T_{\text{base}}^{\text{place}}$. This represents the final position and orientation of the part being manipulated. The place orientation, R^{place}, should be selected in such a manner that the approach vector r^3 is orthogonal to the surface on which the part will come to rest. Furthermore, the distance d^{place} between the place position and the place surface, as measured along the approach vector, should be identical to the distance d^{pick} between the pick position and the pick surface:

$$d^{\text{place}} = d^{\text{pick}} \tag{4-5-2}$$

Indeed, if $d^{\text{place}} < d^{\text{pick}}$, then an attempt will be made to *penetrate* the work surface when the part is placed, as shown at the left in Fig. 4-14. In this case the part will probably *slip* within the jaws of the tool or gripper. Alternatively, if $d^{\text{place}} > d^{\text{pick}}$, the placed part will be unsupported when it is released, as shown at the right in Fig. 4-14; gravity will then cause it to fall to the work surface.

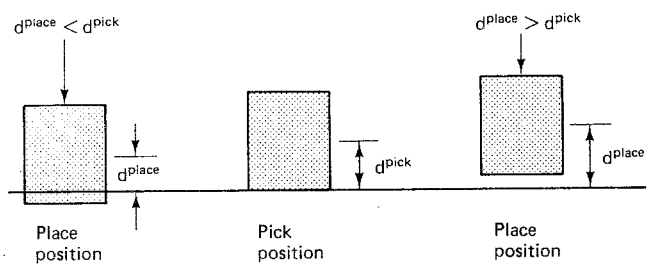


Figure 4-14 The place distance constraint.

The simplest and most important special case of the place position constraint in Eq. (4-5-2) occurs when a part is picked up from and placed down onto a common, horizontal work surface. In this case, Eq. (4-5-2) reduces to $p_3^{\text{place}} = p_3^{\text{pick}}$. More generally, if parts are being *stacked* vertically and the vertical separation between the pick surface and the place surface is the height of the stack, denoted h^{stack}, then the following constraint must be satisfied for a proper stacking operation:

$$p_3^{\text{place}} = h^{\text{stack}} + p_3^{\text{pick}} \tag{4-5-3}$$

$$h^{\text{stack}} = h^{\text{stack}} + 2p_3^{\text{pick}} \tag{4-5-4}$$

Note that the height of the stack is "updated" in Eq. (4-5-4) in anticipation of placing subsequent parts on the stack. It is assumed in Eq. (4-5-4) that the part has been grasped at its centroid, so that the height of the part is $2p_3^{\text{pick}}$. To start the block stacking operation, the height of the stack is initialized to $h^{\text{stack}} = 0$.

The last point of the four-point pick-and-place trajectory is the *set-down point*, whose coordinate frame is denoted $T_{\text{base}}^{\text{set}}$. The set-down point is analogous to the lift-off point. It is a point *near* the place point that the robot moves to before an attempt is made to place the part. Thus a gross motion to the general vicinity of the place location is made first, followed by a fine motion during which the part is carefully placed on the place surface. Whereas the place point is explicitly specified by the user, the set-down point is an intermediate point that is *inferred* from the place point. Once the place-point frame $T_{\text{base}}^{\text{place}}$ has been determined, the associated set-down point frame can be computed as follows:

$$T_{\text{base}}^{\text{set}} = \left[\begin{array}{c|c} R^{\text{place}} & p^{\text{place}} - \nu R^{\text{place}} i^3 \\ \hline 0 \quad 0 \quad 0 & 1 \end{array}\right] \tag{4-5-5}$$

Again, the tool orientation at the set-down point is the same as it is at the place point; that is, $R^{\text{set}} = R^{\text{place}}$. This way the tool orientation remains fixed during the fine motion used to set the part down. The tool position at the set-down point is obtained by starting at the place position and moving backward a distance ν along the approach vector. Thus the set-down point will be a distance of ν away from the place point.

The sequence of motions needed to execute a typical pick-and-place operation is shown in Fig. 4-15. Here a single horizontal work surface is assumed. If part feeders, conveyors, or carousels are used to present parts to the robot or transport parts from the robot, then the pick-and-place surfaces might be at different levels or at different orientations.

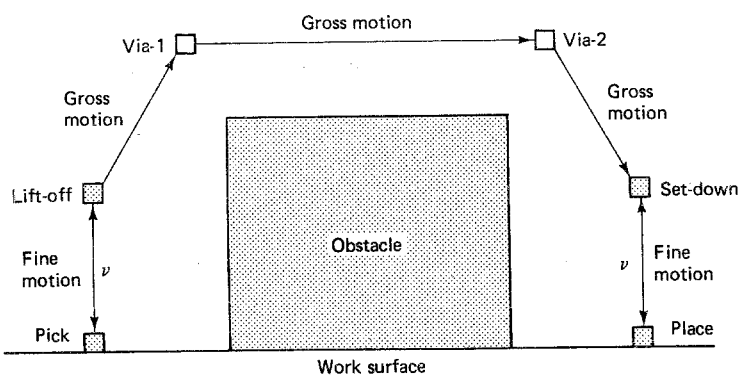


Figure 4-15 A pick-and-place trajectory.

The minimal four-point pick-and-place trajectory is shown with the shaded blocks in Fig. 4-15. If the workspace contains obstacles, then to avoid collisions one or more *via-points* may have to be inserted between the lift-off and set-down points,

as shown in Fig. 4-15. The gross motion from the lift-off point to the set-down point is then traversed by way of a series of obstacle-avoiding via-points.

4-5-3 Speed Variation

The robot controller may be capable of *gross* speed control in the sense that the entire motion of the manipulator can be slowed down or speeded up under software control (Schilling and Williams, 1987). Given this limited form of speed control, the execution of the pick-and-place operation can be made smoother and more reliable by varying the robot speed. For example, the gross motion between the lift-off and set-down points can usually be carried out at high speed for efficiency. However, the fine motions near the two end points of the trajectory should be performed at reduced speed.

The two motions that are the most critical are the movement from the lift-off point to the pick point and the movement from the set-down point to the place point. Note that in both cases the robot is approaching the work surface. Loading on the robotic arm due to gravity will often tend to increase the speed of these movements, particularly when the approach vector is $r^3 = -i^3$. This increase in speed can sometimes cause an *overshoot* of the pick or place point. When the tool moves from the lift-off point to the pick point, an overshoot followed by damped oscillations about the pick point can cause the part to be knocked over or otherwise disturbed. An overshoot when the tool moves from the set-down point to the place point is even more troublesome, because here *any* overshoot causes the robot to attempt to penetrate the work surface with the part.

In the pick-and-place trajectory shown in Fig. 4-15, some of the points, such as the lift-off and set-down points, are visited twice while other points are visited only once. For a typical pick-and-place operation, the proper sequence of points, the types of motion between points, and the relative speeds over various segments of the trajectory are summarized in Table 4-1.

TABLE 4-1 A PICK-AND-PLACE SEQUENCE

Destination	Motion Type	Speed
Lift-off	Gross	Fast
Pause	Fine	Zero
Pick	Fine	Very slow
Grasp	Fine	Slow
Lift-off	Fine	Slow
Via	Gross	Fast
Set-down	Gross	Fast
Pause	Fine	Zero
Place	Fine	Very slow
Release	Fine	Slow
Set-down	Fine	Slow

Note that brief *pauses* in the motion have been inserted prior to the fine motions used to approach the work surface. The high-speed gross motions used to move

the tool to the lift-off and set-down points often generate vibrations caused by an overshoot of the destination. The brief pauses are inserted to allow time for these small oscillations, sometimes called *ringing*, to damp out. The moves immediately following the pauses are the most delicate segments of the pick-and-place operation and should be executed with some care.

There are at least two instances when high-speed motion would be inappropriate during *any* part of a pick-and-place trajectory. One occurs when the part being moved consists of a container filled with a liquid that might spill or otherwise become disturbed as a result of large accelerations. Another instance occurs in a clean room environment. Here rapid motions of the robotic arm can create turbulences in the airflow and therefore cause airborne contaminants to be deposited on the part. These contaminants can generate defects which ultimately reduce productivity by lowering yield. Consequently, this is an instance where increasing the robot speed does *not* increase the product throughput!

4-6 CONTINUOUS-PATH MOTION

If the speed of each joint of the robot can be controlled *independently*, then the robotic manipulator is capable of *continuous-path* motion control as opposed to the more primitive *point-to-point* motion control. Continuous-path motion is required in those applications where the tool must follow a specific path between points in the workspace. Typically, the speed with which the tool moves along that path must also be regulated. Examples of applications in which continuous-path motion control is important include paint spraying, arc welding, and the application of glue or sealant.

4-6-1 Paths and Trajectories

The path-planning problem for continuous-path motion control is naturally formulated in tool configuration space $\mathbf{R}^6$. Recall that the tool-configuration vector $w = (w^1, w^2)$ is a six-dimensional representation of tool tip position w^1 and tool orientation w^2, defined as follows:

$$w^1(q) = p(q) \tag{4-6-1}$$

$$w^2(q) = [\exp(q_n/\pi)]r^3(q) \tag{4-6-2}$$

Thus, a tool path can be represented as a curve in tool-configuration space $\mathbf{R}^6$. A tool path is a purely *spatial* representation. If *temporal* information is added by specifying the times at which the tool must be at various points along the path, then the path becomes a *trajectory*. We separate the spatial and temporal aspects of a tool trajectory by defining the trajectory in a parametric manner as follows:

Definition 4-6-1: Tool Trajectory. Let Γ be a curve in tool-configuration space $\mathbf{R}^6$ and let $s(t)$ be a differentiable *speed distribution function* mapping $[0, T]$ into $[0, 1]$, with $s(0) = 0$ and $s(T) = 1$. Then the *tool trajectory* associated with $\{\Gamma, s\}$ is defined:

$$\Gamma \triangleq \{w(\lambda) \in \mathbf{R}^6 : 0 \leq \lambda \leq 1\}$$
$$\lambda \triangleq s(t) \qquad 0 \leq t \leq T$$

Here the curve Γ in $\mathbf{R}^6$ represents the path that the tool is to follow. The parameter λ is a normalized *path length* parameter where $\lambda = 0$ corresponds to the start of the path and $\lambda = 1$ corresponds to the end of the path. The speed at which the tool moves along the path is specified by the speed distribution function $s(t)$. The simplest example of a speed distribution function which satisfies the two end-point constraints is:

$$s(t) = \frac{t}{T} \qquad 0 \leq t \leq T \tag{4-6-3}$$

Here $T > 0$ is the total time required to traverse the path. The time derivative of the speed distribution function, $\dot{s}(t)$, represents the *instantaneous speed* of the tool tip at time t. The special case in Eq. (4-6-3) therefore corresponds to a *constant* tool speed of $1/T$. A plot of $\dot{s}(t)$ versus t is called a *speed profile*. A typical example of a speed profile is shown in Fig. 4-16. Here the robot ramps up, or accelerates, to a running speed at the start of the trajectory, $[0, \tau]$, and then ramps back down, or decelerates, to zero speed at the end of the trajectory, $[T - \tau, T]$. This ramping up and down effectively limits the maximum acceleration required.

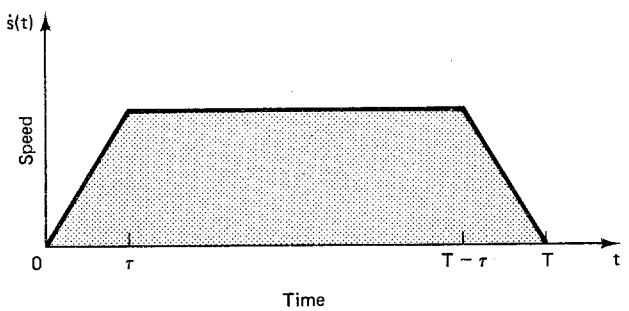


Figure 4-16 A typical speed profile.

Because $s(0) = 0$ and $s(T) = 1$, it follows from the fundamental theorem of calculus that the area under the speed profile curve will always be unity. The overall speed of the tool tip can be varied by changing T, while the instantaneous speed at any point along the trajectory is varied by changing the shape or slope of $s(t)$. Note that if $\dot{s}(t) \geq 0$ for $0 \leq t \leq T$, then the tool will never move backward, and if $\dot{s}(t) > 0$ for $0 \leq t \leq T$, it will also not pause at any point along the path.

Given a path Γ in tool-configuration space and a speed distribution function $s(t)$ for moving along that path, it remains to determine the *joint rates* $\dot{q}$ that will generate the desired tool trajectory. We must compute the rate of change of the joint variables because the robot is controlled directly at the joint level rather than at the tool level. The kth joint velocity $\dot{q}_k$ is directly proportional to the output speed of the kth motor or actuator for $1 \leq k \leq n$. The most direct method for determining the required joint rates is to simply differentiate the closed-form inverse kinematics equations, assuming they are available. Since the details of this approach are robot-dependent, we will illustrate it with specific classes of robots. In Chap. 5, we exam-

ine an alternative approach called *resolved-motion rate control* that is applicable to robotic manipulators in general.

4-6-2 Continuous-Path Control of a Five-Axis Articulated Robot (Rhino XR-3)

Consider the case of a five-axis articulated robot with tool pitch and tool roll motion as in Fig. 4-1. From Algorithm 3-4-1, the inverse kinematics solution for the joint variables q given a tool-configuration vector w is:

$$q_1 = \text{atan2}\ (w_2, w_1) \tag{4-6-4}$$

$$q_3 = \pm\arccos \frac{b_1^2 + b_2^2 - a_2^2 - a_3^2}{2a_2a_3} \tag{4-6-5}$$

$$q_2 = \text{atan2}\ [(a_2 + a_3C_3)b_2 - a_3S_3b_1,\ (a_2 + a_3C_3)b_1 + a_3S_3b_2] \tag{4-6-6}$$

$$q_4 = q_{234} - q_2 - q_3 \tag{4-6-7}$$

$$q_5 = \pi \ln (w_4^2 + w_5^2 + w_6^2)^{1/2} \tag{4-6-8}$$

Here the intermediate variables $\{b_1, b_2, q_{234}\}$ are obtained from q_1 and from the tool-configuration vector w as follows, where q_{234} represents the global tool pitch angle:

$$b_0 = C_1w_4 + S_1w_5 \tag{4-6-9}$$

$$q_{234} = \text{atan2}\ (-b_0, -w_6) \tag{4-6-10}$$

$$b_1 = C_1w_1 + S_1w_2 - a_4C_{234} + d_5S_{234} \tag{4-6-11}$$

$$b_2 = d_1 - a_4S_{234} - d_5C_{234} - w_3 \tag{4-6-12}$$

Base rate: $\dot{q}_1$. The easiest joint rate to extract is the rate at which the base angle q_1 should be driven. Differentiating Eq. (4-6-4), and simplifying, we find that:

$$\dot{q}_1 = \frac{w_1\dot{w}_2 - w_2\dot{w}_1}{w_1^2 + w_2^2} \tag{4-6-13}$$

Note that if $w_2(t) = \gamma w_1(t)$ for some constant γ, then $\dot{q}_1(t) = 0$. This is as it should be, because if w_2 is proportional to w_1, then the arm is moving radially, which means that there is no angular motion about the base.

Elbow rate: $\dot{q}_3$. The joint rate for the elbow joint is computationally more involved. This is due to the coupling that exists between the elbow, shoulder, and tool pitch joints in a vertical-jointed robot. From Eq. (4-6-5), we see that the elbow joint depends on the intermediate variables $\{b_1, b_2\}$. Differentiating Eq. (4-6-9) through (4-6-12), we find that the rates at which the intermediate variables are changing are as follows:

$$\dot{b}_0 = C_1\dot{w}_4 + S_1\dot{w}_5 + (C_1w_5 - S_1w_4)\dot{q}_1 \tag{4-6-14}$$

$$\dot{q}_{234} = \frac{w_6\dot{b}_0 - b_0\dot{w}_6}{w_6^2 + b_0^2} \tag{4-6-15}$$

$$\dot{b}_1 = C_1\dot{w}_1 + S_1\dot{w}_2 + (C_1w_2 - S_1w_1)\dot{q}_1 + (a_4S_{234} + d_5C_{234})\dot{q}_{234} \quad (4\text{-}6\text{-}16)$$

$$\dot{b}_2 = (d_5S_{234} - a_4C_{234})\dot{q}_{234} - \dot{w}_3 \quad (4\text{-}6\text{-}17)$$

The rate at which the elbow joint should be driven can now be expressed in terms of $\{b_1, b_2\}$ and their derivatives. Differentiating Eq. (4-6-5) and simplifying, we have:

$$\dot{q}_3 = \mp \frac{2(b_1\dot{b}_1 + b_2\dot{b}_2)}{[(2a_2a_3)^2 - (b_1^2 + b_2^2 - a_2^2 - a_3^2)^2]^{1/2}} \quad (4\text{-}6\text{-}18)$$

Shoulder rate: $\dot{q}_2$. The shoulder rate is, computationally, the most expensive joint rate to determine. To simplify the final expression for the shoulder rate, it is helpful to introduce two additional intermediate variables $\{b_3, b_4\}$ and to evaluate their derivatives:

$$b_3 = (a_2 + a_3C_3)b_1 + a_3S_3b_2 \quad (4\text{-}6\text{-}19)$$

$$b_4 = (a_2 + a_3C_3)b_2 - a_3S_3b_1 \quad (4\text{-}6\text{-}20)$$

$$\dot{b}_3 = (a_2 + a_3C_3)\dot{b}_1 + a_3S_3\dot{b}_2 + a_3(C_3b_2 - S_3b_1)\dot{q}_3 \quad (4\text{-}6\text{-}21)$$

$$\dot{b}_4 = (a_2 + a_3C_3)\dot{b}_2 - a_3S_3\dot{b}_1 - a_3(C_3b_1 + S_3b_2)\dot{q}_3 \quad (4\text{-}6\text{-}22)$$

From inspection of Eq. (4-6-6), we see that $q_2 = \text{atan2}\ (b_4, b_3)$. Thus the rate at which the shoulder joint should be driven can be expressed in terms of $\{b_3, b_4\}$ and their derivatives as follows:

$$\dot{q}_2 = \frac{b_3\dot{b}_4 - b_4\dot{b}_3}{b_3^2 + b_4^2} \quad (4\text{-}6\text{-}23)$$

Note that the final expression for the shoulder rate is of the same general form as the expression for the base rate, because both joint variables are obtained from the atan2 function.

Tool pitch rate: $\dot{q}_4$. The work for determining the rate at which the tool pitch joint should be driven is already in place. Given the global tool pitch rate $\dot{q}_{234}$ in Eq. (4-6-15), the shoulder rate $\dot{q}_2$, and the elbow rate $\dot{q}_3$, we observe from Eq. (4-6-7) that the required tool pitch rate is simply:

$$\dot{q}_4 = \dot{q}_{234} - \dot{q}_2 - \dot{q}_3 \quad (4\text{-}6\text{-}24)$$

Tool roll rate: $\dot{q}_5$. The rate at which the tool roll joint should be driven can be obtained directly from the last three components of the tool-configuration vector w. Differentiating Eq. (4-6-8) and simplifying, we have:

$$\dot{q}_5 = \frac{\pi(w_4\dot{w}_4 + w_5\dot{w}_5 + w_6\dot{w}_6)}{w_4^2 + w_5^2 + w_6^2} \quad (4\text{-}6\text{-}25)$$

Exercise 4-6-1: Vertical Motion. Suppose the tool is required to follow a vertical path defined by the following trajectory in tool-configuration space:

$$w(t) = \left[a_3 + a_4,\ 0,\ (d_1 + a_2 - d_5)\left(1 - \frac{t}{T}\right),\ 0,\ 0,\ -1\right]^T \qquad 0 \le t \le T$$

Find $q(0)$ and $\dot{q}(t)$ for $0 \le t \le T$.

4-6-3 Continuous-Path Control of a Four-Axis SCARA Robot (Adept One)

As a second example of rate equations for continuous-path motion, consider a four-axis SCARA robot with tool roll motion as in Fig. 4-3. From Algorithm 3-5-1, the inverse kinematics solution for the joint variables q given a tool-configuration vector w is:

$$q_2 = \pm\arccos\frac{w_1^2 + w_2^2 - a_1^2 - a_2^2}{2a_1a_2} \tag{4-6-26}$$

$$q_1 = \text{atan2}\ (b_2, b_1) \tag{4-6-27}$$

$$q_3 = d_1 - d_4 - w_3 \tag{4-6-28}$$

$$q_4 = \pi \ln |w_6| \tag{4-6-29}$$

Here the intermediate variables $\{b_1, b_2\}$ used in Eq. (4-6-27) are obtained from q_2 and the tool-configuration vector w as follows:

$$b_1 = (a_1 + a_2C_2)w_1 - a_2S_2w_2 \tag{4-6-30}$$

$$b_2 = (a_1 + a_2C_2)w_2 + a_2S_2w_1 \tag{4-6-31}$$

Elbow rate: $\dot{q}_2$. The first two joints are coupled in a horizontal-jointed or SCARA robot. The simplest rate to extract in this case is the rate at which the elbow angle q_2 should be driven. Differentiating Eq. (4-6-26) and simplifying, we find that:

$$\dot{q}_2 = \mp\frac{2(w_1\dot{w}_1 + w_2\dot{w}_2)}{[(2a_1a_2)^2 - (w_1^2 + w_2^2 - a_1^2 - a_2^2)^2]^{1/2}} \tag{4-6-32}$$

Base rate: $\dot{q}_1$. The joint rate for the base joint is computationally more involved. From Eq. (4-6-27), we see that the base joint depends on the intermediate variables $\{b_1, b_2\}$. Differentiating Eqs. (4-6-30) and (4-6-31), we find that the rates at which the intermediate variables are changing are as follows:

$$\dot{b}_1 = (a_1 + a_2C_2)\dot{w}_1 - a_2S_2\dot{w}_2 - a_2(S_2w_1 + C_2w_2)\dot{q}_2 \tag{4-6-33}$$

$$\dot{b}_2 = (a_1 + a_2C_2)\dot{w}_2 + a_2S_2\dot{w}_1 + a_2(C_2w_1 - S_2w_2)\dot{q}_2 \tag{4-6-34}$$

The rate at which the base joint should be driven can now be expressed in terms of $\{b_1, b_2\}$ and their derivatives. Differentiating Eq. (4-6-27) and simplifying, we have:

$$\dot{q}_1 = \frac{b_1\dot{b}_2 - b_2\dot{b}_1}{b_1^2 + b_2^2} \tag{4-6-35}$$

Vertical extension rate: $\dot{q}_3$. The easiest joint rate to obtain for a SCARA robot is the rate at which the prismatic vertical extension joint should be driven. This joint is completely uncoupled from the other three. Differentiating Eq. (4-6-28), we get:

$$\dot{q}_3 = -\dot{w}_3 \tag{4-6-36}$$

Tool roll rate: $\dot{q}_4$. The rate at which the tool roll joint should be driven can be obtained directly from the last component of the tool configuration vector w. Differentiating Eq. (4-6-29) and simplifying, we have the following expression for $\dot{q}_4$:

$$\dot{q}_4 = \frac{\pi \dot{w}_6}{w_6} \tag{4-6-37}$$

Exercise 4-6-2: Threading Motion. Suppose the desired tool path for a SCARA robot is a vertical threading motion defined by the following trajectory in tool-configuration space:

$$w(t) = \left[a_1,\ a_2,\ (d_1 - d_4)\left(1 - \frac{t}{T}\right),\ 0,\ 0,\ -\exp\left(\frac{2t}{T} - 1\right)\right]^T \qquad 0 \le t \le T$$

Find $q(0)$ and $\dot{q}(t)$ for $0 \le t \le T$.

Exercise 4-6-3: Three-Axis Planar Articulated Robot. Let $w(t)$ be a differentiable tool path for the three-axis planar articulated robot whose inverse kinematics are given in Algorithm 3-7-1. Derive an expression for the joint rate $\dot{q}(t)$ in terms of $w(t)$ and $\dot{w}(t)$.

4-7 INTERPOLATED MOTION

In the previous section we assumed that the path to be followed by the tool was specified explicitly and completely by a curve Γ in tool-configuration space $\mathbf{R}^6$. In many instances, the path will not be completely specified. Instead, *knot points* along the path, such as the end points and perhaps intermediate via-points, will be specified. In these circumstances, it is the responsibility of the trajectory-planning software to *interpolate* between the knot points to produce a smooth trajectory Γ that can be executed using continuous-path motion control techniques. For the general interpolation problem, we have a sequence of m knot points in tool-configuration space that must be *visited* by the tool:

$$\Gamma_m = \{w^0, \ldots, w^{m-1}\} \tag{4-7-1}$$

The path $w(t)$ between the m knot points should be *smooth*. That is, $w(t)$ should have at least two continuous derivatives in order to avoid the need for infinite acceleration. In practice, short bursts of high acceleration manifest themselves as jerky robot motion, motion that can burn out drive circuits and strip gears.

4-7-1 Cubic Polynomial Paths

We begin the analysis of interpolated motion with the simplest special case, $m = 2$. As a candidate for interpolating a smooth path between two points $\{w^0, w^1\}$, consider the following cubic polynomial path:

$$w(t) = at^3 + bt^2 + ct + d \qquad 0 \le t \le T \tag{4-7-2}$$

Here the constant $T > 0$ is the time required to traverse the path. Since Eq. (4-7-2) represents a trajectory in tool configuration space, the polynomial coefficients $\{a, b, c, d\}$ are vectors in $\mathbf{R}^6$. The two position constraints at the end points of the trajectory are $w(0) = w^0$ and $w(T) = w^1$. These constraints remove two of the four degrees of freedom available for choosing the coefficients of $w(t)$. The remaining two degrees of freedom can be used to specify the velocities at the end points of the trajectory as follows:

Proposition 4-7-1: Cubic Polynomial Interpolation. Suppose the robot is to move from point w^0 to point w^1 in tool-configuration space over an interval of time $[0, T]$. If the tool is to start out at time 0 with velocity v^0 and end up at time T with velocity v^1, then the cubic polynomial trajectory in Eq. (4-7-2) can be used to interpolate between the points with:

$$a = \frac{T(v^1 + v^0) - 2(w^1 - w^0)}{T^3}$$

$$b = -\frac{[T(v^1 + 2v^0) - 3(w^1 - w^0)]}{T^2}$$

$$c = v^0$$

$$d = w^0$$

Proof. There are four constraints on the four unknown coefficients of the cubic interpolating polynomial. The two end-point position constraints are $w(0) = w^0$ and $w(T) = w^1$. Thus, from Eq. (4-7-2), we have:

$$d = w^0$$

$$aT^3 + bT^2 + cT + d = w^1$$

If the tool is to start with velocity v^0 at the beginning of the trajectory and end up with velocity v^1 at the end of the trajectory, then the two end-point velocity constraints are $\dot{w}(0) = v^0$ and $\dot{w}(T) = v^1$. Thus from Eq. (4-7-2), we have:

$$c = v^0$$

$$3aT^2 + 2bT + c = v^1$$

From inspection, $d = w^0$ and $c = v^0$. Solving the remaining two equations for $\{a, b\}$ then yields the coefficients specified in Prop. 4-7-1.

In the event that the discrete path Γ_m has more than two knot points, the curve-fitting problem begins to become more involved. One approach is to simply use a higher-degree interpolating polynomial. The increase in the number of coefficients provides more degrees of freedom for satisfying additional position and velocity constraints. The problem with using higher-degree polynomials is that they typically generate paths that have large oscillations *between* the knot points.

Another approach is to successively apply Prop. 4-7-1 to each two-point segment. Although this is computationally very simple, it has the drawback that the user must explicitly *specify* values for the via-point velocities (Craig, 1986). Of course, zero velocities could be used, but then the tool *pauses* at each knot point. Yet another approach is to take adjacent cubic polynomial segments and *spline* them together by requiring that the velocity and the acceleration be *continuous* at the segment boundaries. This frees the user from having to specify velocities at the via-points. The difficulty encountered here is that the equations for computing the polynomial coefficients for the segments become *coupled*. For example, for an m-point trajectory the total number of equations that have to be solved is $4(m-1)$. Thus, for $m > 2$, it becomes difficult to obtain simple closed-form expressions for the polynomial coefficients.

4-7-2 Linear Interpolation with Parabolic Blends

An alternative to cubic polynomial interpolation is to use *piecewise-linear* interpolation between the knot points as shown in Fig. 4-17. This effectively decouples the original m-point problem into $m-1$ separate two-point problems.

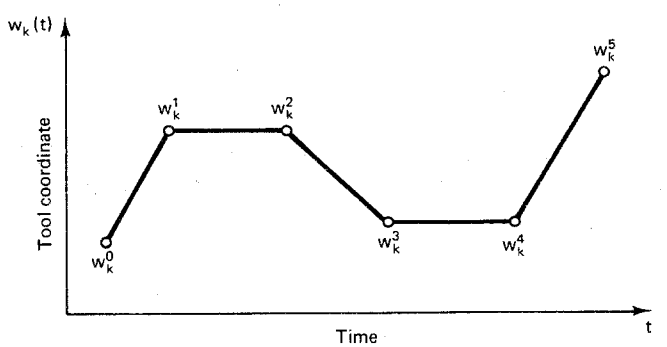


Figure 4-17 Piecewise-linear interpolation between knot points, $1 \leq k \leq 6$.

Although piecewise-linear interpolation is computationally very efficient, it suffers from one major drawback: the path it generates is not smooth. Indeed, since the velocity changes abruptly as the tool passes through a knot point, an *infinite* instantaneous tool acceleration is required. This drawback can be remedied by introducing *parabolic blends* which smoothly connect adjacent piecewise-linear segments (Taylor, 1979). As an illustration of this technique, consider the two-segment trajec-

tory shown in Fig. 4-18. Suppose that segment $\{w^{k-1}, w^k\}$ is to be traversed in time T_k with constant velocity v^k for $1 \leq k \leq 2$. Then $v^k = \Delta w^k / T_k$, where:

$$\Delta w^k \triangleq w^k - w^{k-1} \qquad 1 \leq k \leq 2 \tag{4-7-3}$$

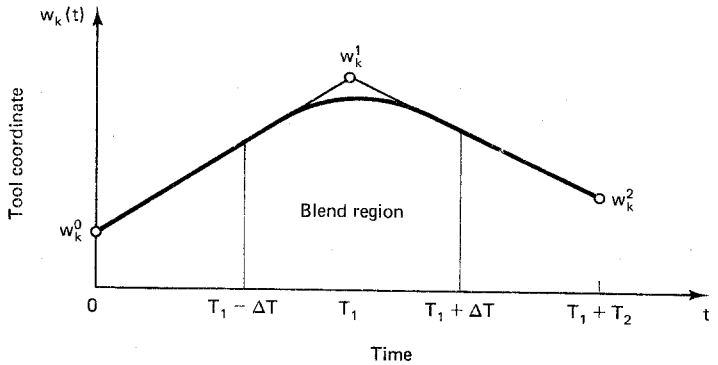


Figure 4-18 Piecewise-linear interpolation with a parabolic blend, $1 \leq k \leq 6$.

Now if $v^2 \neq v^1$, then an instantaneous infinite acceleration is required at time T_1 to achieve the abrupt change in velocity. To keep the acceleration finite, we must *gradually* change the velocity over some *transition interval* $[T_1 - \Delta T, T_1 + \Delta T]$, where $\Delta T > 0$. Here we apply a constant acceleration starting at time $T_1 - \Delta T$. By properly selecting the acceleration, a smooth transition can be achieved from velocity v^1 at time $T_1 - \Delta T$ to velocity v^2 at time $T_1 + \Delta T$. This will place the tool, at time $T_1 + \Delta T$, at the same point where it would have been if the original piecewise-linear path had been followed. Since the acceleration is constant, the tool trajectory during the transition interval $[T_1 - \Delta T, T_1 + \Delta T]$ will be a quadratic polynomial of the general form:

$$w(t) = \frac{a(t - T_1 + \Delta T)^2}{2} + b(t - T_1 + \Delta T) + c \tag{4-7-4}$$

For analysis purposes, suppose that the *transition time* ΔT is known. The required acceleration a can then be found in terms of the transition time ΔT, the segment traversal times $\{T_k\}$, and the segment displacements $\{\Delta w^k\}$, as can be seen from the following result:

Proposition 4-7-2: Parabolic Blends. Let a be the constant acceleration required to smoothly transfer the tool from velocity $\Delta w^1 / T_1$ at time $T_1 - \Delta T$ to velocity $\Delta w^2 / T_2$ at time $T_1 + \Delta T$ on the piecewise-linear trajectory in Fig. 4-18. Then:

$$a = \frac{T_1 \, \Delta w^2 - T_2 \, \Delta w^1}{2 T_1 T_2 \, \Delta T}$$

Proof. Given the quadratic expression in Eq. (4-7-4), if we differentiate it twice we see that the acceleration is constant and equal to a. To evaluate a, we apply

the necessary boundary conditions. The velocity constraint at the start of the transition is $\dot{w}(T_1 - \Delta T) = \Delta w^1/T_1$. Differentiating Eq. (4-7-4) and evaluating the result at $T_1 - \Delta T$ yields:

$$b = \frac{\Delta w^1}{T_1}$$

The velocity constraint at the end of the transition is $\dot{w}(T_1 + \Delta T) = \Delta w^2/T_2$. Differentiating Eq. (4-7-4) and evaluating the result at $T_1 + \Delta T$ yields:

$$2\,\Delta T\,a + b = \frac{\Delta w^2}{T_2}$$

Combining the two equations for a and b and solving for a then yields the expression in Prop. 4-7-2.

The transition time ΔT can be chosen by the user, but the choice is not arbitrary. From Prop. 4-7-2 it is clear that one must have $\Delta T > 0$ for a finite acceleration a. The transition time ΔT is also bounded from above, because otherwise the transition interval $[T_1 - \Delta T, T_1 + \Delta T]$ might not lie inside the overall interval $[0, T_1 + T_2]$. Thus the bounds on the transition time ΔT are:

$$0 < \Delta T < \min\{T_1, T_2\} \tag{4-7-5}$$

More generally, if there are blend regions associated with the points adjacent to w^1, then the transition time ΔT will have to be still smaller so that successive blends do not overlap. The parabolic-blend trajectory does not actually go through the knot point w^1 at time T_1. However, it can be made to go arbitrarily close to the knot point if the transition time ΔT is made small enough, as can be seen from the following exercise:

Exercise 4-7-1: Knot-Point Deviation. Compute the knot-point *deviation* $\Delta w(\Delta T) = w(T_1) - w^1$. Show that $\Delta w(\Delta T) \to 0$ as $\Delta T \to 0$.

As the transition time ΔT becomes smaller, the parabolic-blend trajectory approaches the original piecewise-linear trajectory. However, it is evident from Prop. 4-7-2 that the required acceleration increases as the transition time decreases. In the limit as ΔT approaches zero, the required acceleration is in fact infinite. If the upper bound on ΔT in Eq. (4-7-5) requires an acceleration that exceeds the capability of the robot actuators, then the specified motion cannot be achieved. In this case the motion must be slowed down by increasing the segment traversal times $\{T_1, T_2\}$. For example, if T_1 and T_2 are doubled, then ΔT can also be doubled, in which case the required acceleration a is reduced by a factor of four.

To evaluate the actual tool trajectory followed during the parabolic blend, we must evaluate the remaining coefficient of the quadratic polynomial in Eq. (4-7-4). The coefficients $\{a, b\}$ have already been determined in the proof of Prop. 4-7-2. To evaluate the constant c, we use the position constraint at the start of the transition, namely, $w(T_1 - \Delta T) = w^1 - (\Delta T/T_1)\,\Delta w^1$. From Eq. (4-7-4), we have

$w(T_1 - \Delta T) = c$. Thus the complete tool trajectory followed during a two-segment blended piecewise-linear interpolation is as follows:

Proposition 4-7-3: Blended Trajectory. Let $w(t)$ be a blended piecewise-linear trajectory through the points $\{w^0, w^1, w^2\}$ with segment traversal times of $\{T_k\}$ and a blend transition time of ΔT. Then the blend acceleration a is as in Prop. 4-7-2, and the complete trajectory is:

$$w(t) = \begin{cases} w^0 + \dfrac{t\,\Delta w^1}{T_1} & 0 \le t \le T_1 - \Delta T \\ \dfrac{a}{2}(t - T_1 + \Delta T)^2 + \dfrac{\Delta w^1 (t - T_1)}{T_1} + w^1 & T_1 - \Delta T < t < T_1 + \Delta T \\ w^1 + \dfrac{(t - T_1)\,\Delta w^2}{T_2} & T_1 + \Delta T \le t \le T_1 + T_2 \end{cases} \tag{4-7-6}$$

Exercise 4-7-2: Blended Trajectory. Verify that the blended trajectory in Prop. 4-7-3 is continuous at the two points, $w(T_1 - \Delta T)$ and $w(T_1 + \Delta T)$.

4-8 STRAIGHT-LINE MOTION

Straight-line motion of the robot payload represents the shortest distance between two points in the workspace. There are a variety of applications in which the tool is required to follow, or at least closely approximate, a straight-line path. Examples include tracking a conveyor belt, following a straight-line seam during arc welding, and inserting a peg into a hole. In many of these applications, the straight-line path is traversed at a constant speed. These are instances of *uniform* straight-line motion, where the inertial forces on the payload (but not the links) are minimized.

To plan a trajectory in joint space which generates a straight-line trajectory in tool-configuration space, we can use the inverse kinematics equations. Suppose in particular that w^0 and w^1 denote the *initial point* and *final point*, respectively, in tool-configuration space $\mathbf{R}^6$. If the movement is to be carried out in T seconds, then a general straight-line trajectory for the tool can be represented as follows:

$$w(t) = [1 - s(t)]w^0 + s(t)w^1 \qquad 0 \le t \le T \tag{4-8-1}$$

Here $s(t)$ is a differentiable speed distribution function mapping $[0, T]$ into $[0, 1]$ with $s(0) = 0$ and $s(T) = 1$. Typically the speed profile $\dot{s}(t)$ might ramp up at a constant acceleration to reach a running speed, then proceed at constant speed, and finally ramp down at a constant deceleration to zero. For the case of uniform straight-line motion, the speed distribution function is simply:

$$s(t) = \frac{t}{T} \tag{4-8-2}$$

If the robot is not a general manipulator in the sense of having six axes, then the tool-configuration vector $w(t)$ cannot be made to follow arbitrary straight-line

paths in $\mathbf{R}^6$. However, one important class of straight-line paths that most manipulators can follow corresponds to the case of moving the tool tip p along the line defined by a constant approach vector r^3. This type of motion includes, for example, *insert* and *extract* operations for fastening and unfastening parts. If the linear motion along r^3 is accompanied by angular motion about r^3, then the insert and extract operations become *thread* and *unthread* fastening operations, as in Eq. (4-4-8). For a five-axis articulated robot with tool pitch and tool roll motion, the robot can be directed to move along any approach vector r^3 that is achievable by the robot as long as the tool-tip position remains within the work envelope.

Another important special case of straight-line motion occurs when the approach vector is fixed at $r^3 = -i^3$. In the case of a fixed vertical approach vector, the tool tip p of both the five-axis articulated robot and the four-axis SCARA robot can be made to trace arbitrary straight-line paths within the work envelope.

For many robots, only point-to-point motion control is available, because the speeds of the individual joints cannot be varied independently. Here an alternative to continuous-path motion control must be found to *approximate* straight-line motion. It is not sufficient to simply follow a straight-line path in joint space between the end points, because, in general, this will not produce a straight-line path in tool-configuration space. However, if the distance between adjacent points in joint space is sufficiently small, then a straight-line path segment in joint space will approximate a straight-line path segment in tool-configuration space. Thus we can approximate a straight-line path by visiting an appropriate sequence of closely spaced *knot points* in joint space.

Since the inverse kinematics equations have to be solved at each knot point, it is desirable to minimize the number of knot points and distribute them along the trajectory in an optimal manner. A simple, yet effective, technique for selecting the knot points called the *bounded deviations method* has been proposed by Taylor (1979). The basic premise is the observation that the *deviation* between the actual straight-line trajectory in Eq. (4-8-1) and the trajectory generated by a straight-line motion in joint space is likely to be maximum somewhere near the midpoint of the joint-space trajectory. The deviation or error can be checked at this point, and if it exceeds a prescribed tolerance, then the exact midpoint is added as a knot point. This test is then repeated recursively on the two newly generated segments until all of the midpoint deviations are within the prescribed tolerance. The following is an adaptation of Taylor's algorithm for approximating straight-line motion with a point-to-point robot.

Algorithm 4-8-1: Bounded Deviation

1. Select $\epsilon > 0$. Use the inverse kinematics equations to compute $\{q^0, q^1\}$, joint vectors associated with $\{w^0, w^1\}$.
2. Compute the joint-space midpoint:

$$q^m = \frac{q^0 + q^1}{2}$$

3. Use q^m and the tool-configuration function w to compute the associated tool-configuration space midpcint:

$$w^m = w(q^m)$$

4. Compute the exact tool-configuration space midpoint:

$$w^M = \frac{(w^0 + w^1)}{2}$$

5. If the deviation $\| w^m - w^M \| \leq \epsilon$, then stop; else, insert w^M as a knot point between w^0 and w^1.
6. Apply Algorithm 4-8-1 recursively to the two new segments $\{w^0, w^M\}$ and $\{w^M, w^1\}$.

Note that Algorithm 4-8-1 does not necessarily distribute the knot points uniformly over the interval $\{w^0, w^1\}$. Instead, it places them where they are most needed to reduce the deviation between the exact trajectory and the joint-space interpolated trajectory. An example of an approximation to a straight-line path in tool-configuration space, using a sequence of knot points in joint space, is shown in Fig. 4-19.

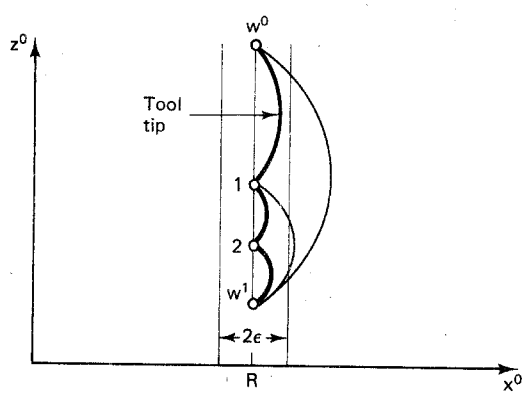


Figure 4-19 Interpolated joint-space approximation to straight-line motion.

Note from Fig. 4-19 that the joint-space interpolated motion between the two end points w^0 and w^1 deviates well outside the cylinder of radius ϵ, and hence knot point 1 is inserted. Next, the joint-space interpolated motion between end point w^0 and knot point 1 does not deviate outside the cylinder of radius ϵ. Consequently, there is no need for an additional knot point between w^0 and knot point 1. Finally, the joint-space interpolated motion between knot point 1 and w^1 again deviates outside the cylinder of radius ϵ, and for this reason, a second knot point is inserted between knot point 1 and w^1. In general, the distribution of knot points will not be uniform and will depend on ϵ, the robot geometry, and the location of the straight-line path within the workspace.

Example 4-8-1: SCARA Robot

As an example of an application of one pass of Algorithm 4-8-1, consider the SCARA

robot in Fig. 4-3. For simplicity, we assume $a_2 = a_1/2$. Suppose the two end points for the desired straight-line trajectory are:

$$w^0 = [a_1, 0, d_1 - d_4, 0, 0, -1]^T$$

$$w^1 = [a_1 + a_2, 0, d_1 - d_4, 0, 0, -1]^T$$

In this case the tool is extending radially along the x-axis a distance $a_2 = a_1/2$. The equivalent discrete two-point trajectory in joint space is computed by applying the inverse kinematics equations summarized in Algorithm 3-5-1. The result is:

$$q^0 = [0.505, 1.823, 0, 0]^T$$

$$q^1 = [0, 0, 0, 0]^T$$

The midpoint of the joint-space trajectory is $q^m = [0.253, 0.912, 0, 0]^T$. Using the tool-configuration function specified in Eq. (4-3-1), the associated tool-configuration space midpoint is:

$$w^m = [1.363a_1, -0.056a_1, d_1 - d_4, 0, 0, -1]^T$$

The exact tool-configuration space midpoint in this case is $w^M = [1.25a_1, 0, d_1 - d_4, 0, 0, -1]^T$. Thus the midpoint deviation between the exact trajectory and the joint-space interpolated trajectory is:

$$\|w^m - w^M\| = 0.126a_1$$

If $0.126a_1$ exceeds the tolerance ϵ, then the new knot point w^M must be added. Algorithm 4-8-1 should then be applied to the trajectory $\{w^0, w^M\}$ and to the trajectory $\{w^M, w^1\}$.

4-9 PROBLEMS

4-1. Consider the link-coordinate diagram for a four-axis cylindrical-coordinate robot with tool roll motion shown in Fig. 4-20. Here the vector of joint variables is $q = [\theta_1, d_2, d_3, \theta_4]^T$. Suppose the joint-space work envelope Q is as follows:

$$\begin{bmatrix} -\pi \\ h \\ r \\ -\pi \end{bmatrix} \leq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} q \leq \begin{bmatrix} \pi \\ H \\ R \\ \pi \end{bmatrix}$$

Find the work envelope Y and the volume of Y.

4-2. Consider a two-axis planar articulated robot with link lengths a_1 and a_2. Suppose both joint angles vary over the complete range $[-\pi, \pi]$.

(a) Find the total work envelope Y and its area when $a_2 = a_1$.

(b) Find the dexterous work envelope Y_d and its area when $a_2 = a_1$.

(c) Find the total work envelope Y and its area when $a_2 < a_1$.

(d) Find the dexterous work envelope Y_d and its area when $a_2 < a_1$.

4-3. Consider the link-coordinate diagram for a five-axis spherical-coordinate robot with tool pitch and tool roll motion shown in Fig. 4-21. Here the vector of joint variables is $q = [\theta_1, \theta_2, d_3, \theta_4, \theta_5]^T$. Suppose the joint-space work envelope of this robot is as follows:

$$\begin{bmatrix} -\pi/2 \\ 0 \\ r \\ -\pi \\ -\pi \\ -\pi/2 \end{bmatrix} \leq \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} q \leq \begin{bmatrix} \pi/2 \\ 3\pi/4 \\ R \\ -\pi/2 \\ \pi \\ 0 \end{bmatrix}$$

Find the reach constraint which specifies the outer surface of the work envelope Y, assuming q_{24} has been specified.

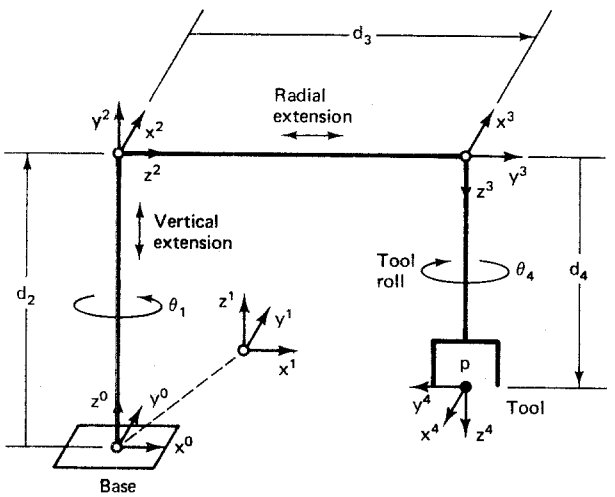


Figure 4-20 Link coordinates of a four-axis cylindrical robot.

4-4. For the spherical-coordinate robot in Fig. 4-21, find bounds on the tool tip coordinates p_1, p_2, and p_3, in that order, assuming the global tool pitch angle q_{24} has been specified.

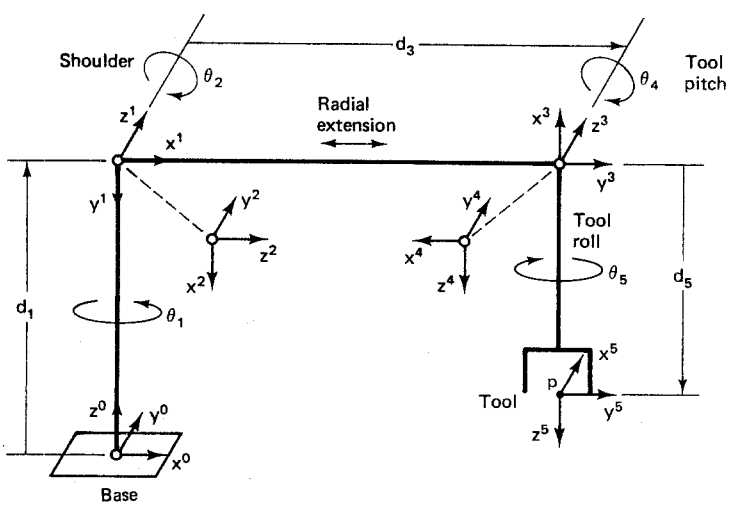


Figure 4-21 Link coordinates of a five-axis spherical robot.

4-5. Consider the top view of a robotic workstation, with parts A and B, shown in Fig. 4-22. Suppose the centroid of part A has coordinates $[6, 12, 2]^T$ and the centroid of part B has coordinates $[10, 5, 1]^T$.

(a) Find the arm matrix value $T_{\text{base}}^{\text{pick}}$ needed to pick up part A from above grasping it along the long sides.

(b) Find the arm matrix value $T_{\text{base}}^{\text{place}}$ needed to place part A on top of part B aligning the centroids and the major axes.

(c) Find the arm matrix value $T_{\text{base}}^{\text{pick}}$ needed to pick up part B from above grasping it along the long sides.

(d) Find the arm matrix value $T_{\text{base}}^{\text{place}}$ needed to place part B on top of part A aligning the centroids and the major axes.

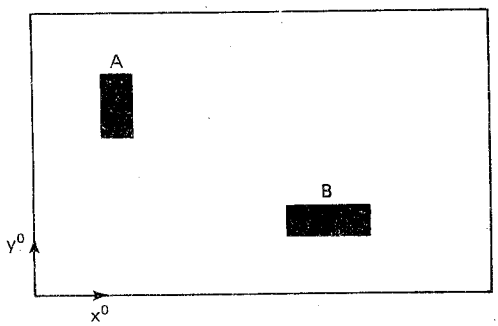


Figure 4-22 Robotic work station.

4-6. Consider the conveyor system shown in Fig. 4-23. Suppose the height of the conveyor belt is 300 mm, the cubes on the conveyor belt are of dimension 50 mm, the belt is moving at a rate of 120 mm/sec., and it is located a distance of 450 mm in front of the robot.

(a) Find an expression for the arm matrix $T_{\text{base}}^{\text{tool}}(t)$ which will allow the tool to track a part on the belt from position $x = -100$ mm to position $x = 0$ while grasping the part on the front and back faces.

(b) Find an expression for the arm matrix $T_{\text{base}}^{\text{tool}}(t)$ which will allow the tool to pick a part up off the conveyor belt starting at $x = 0$ and ending at $x = 100$ mm with a vertical motion of 80 mm/sec. The horizontal motion of the tool should match the speed of the conveyor.

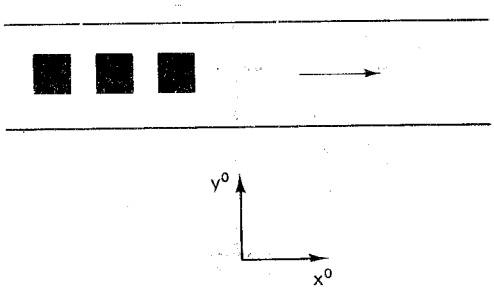


Figure 4-23 Conveyor system.

4-7. Find the cubic interpolating polynomial $w(t)$ which will smoothly move the robot between the following two points in tool-configuration space over the time interval $[0, T]$. You can assume that the velocity is zero at each end of the trajectory.

$$w^0 = [10, 10, 10, 0.368, 0.368, -0.368]^T$$

$$w^1 = [15, 15, 5, 1, 1, -1]^T$$

4-8. Consider a three-point interpolation problem with points $\{w^0, w^1, w^2\}$. Suppose cubic polynomials $u(t)$ and $v(t)$ are used for segments $\{w^0, w^1\}$ and $\{w^1, w^2\}$, respectively, where:

$$u(t) = at^3 + bt^2 + ct + d \qquad 0 \le t < T_1$$

$$v(t) = e(t - T_1)^3 + f(t - T_1)^2 + g(t - T_1) + h \qquad T_1 \le t \le T_1 + T_2$$

The two cubic polynomials are to be splined together at the segment boundary by constraining the velocity and acceleration to be continuous at time T_1. Assuming the velocities at the end points are zero, find the equation $Ax = y$ which represents the constraints on the polynomial coefficients. Here $x = [a, b, c, d, e, f, g, h]^T$ is the vector of unknown coefficients and A and y are to be determined.

4-9. Consider the problem of performing a piecewise-linear interpolation with a parabolic blend on the following three-point trajectory. Here $T_1 = 1$, $T_2 = 1$, and $\Delta T = 0.5$.

$$w^0 = [10, 0, 10, 0, 0, -0.368]^T$$

$$w^1 = [10, 10, 10, 0, 0, -1]^T$$

$$w^2 = [0, 10, 10, 0, 0, -0.368]^T$$

(a) Find the required acceleration a during the blend.
(b) Find the trajectory $w(t)$ over the interval $[0, 2]$.
(c) Find the midpoint deviation $\Delta w(\Delta T) = w(1) - w^1$.

4-10. Suppose the four-axis SCARA robot in Fig. 4-3 is to execute the following trajectory in tool-configuration space:

$$w(t) = [a_1, a_2, d_1 - d_4 - t, 0, 0, -1]^T \qquad 0 \le t \le T$$

Find $q(0)$ and $\dot{q}(:)$. Find a bound on T which prevents the tool from striking the work surface.

4-11. Consider the five-axis articulated robot in Fig. 4-1. Apply one pass of Algorithm 4-8-1 to find a joint-space knot point for approximating the following straight-line trajectory:

$$w^0 = [300, 0, 125, 0, 0, -1]^T$$

$$w^1 = [300, 0, 25, 0, 0, -1]^T$$

What is the tool-configuration deviation at the joint-space midpoint, assuming that $d = [247.6, 0, 0, 0, 193.4]^T$ and $a = [0, 228.6, 228.6, 9.5, 0]^T$?

REFERENCES

BENI, G., and S. HACKWOOD (1985). *Recent Advances in Robotics,* Wiley-Interscience: New York.

BOYES, W. E. (1985). *Low-Cost Jigs, Fixtures, and Gages for Limited Production,* SME Publications: Dearborn, Mich.

CRAIG, J. (1986). *Introduction to Robotics: Mechanics and Control,* Addison-Wesley: Reading, Mass.

SCHILLING, R. J., and S. WILLIAMS (1987). "A firmware voltage controller for a robotic arm," *IEEE Trans. Education,* Vol. E-30, No. 3, pp. 164–173.

TAYLOR, R. H. (1979). "Planning and execution of straight line manipulator trajectories," *IBM J. Research and Development,* Vol. 23, pp. 424–436.

5

Differential Motion and Statics

Once a robot trajectory is planned in tool-configuration space, one must then address the problem of commanding the robot to follow that trajectory. One approach is to differentiate the direct kinematics equations to produce an expression for the tool-configuration velocity in terms of the joint velocity. The transformation from tool-configuration velocity to joint velocity is called the tool-configuration Jacobian matrix. At certain locations in joint space the tool-configuration Jacobian matrix loses rank, and these points are referred to as *singular* configurations of the arm. Since the tool-configuration Jacobian matrix is often not a square matrix, a generalized inverse of it must be computed in order to solve for joint velocity in terms of the desired tool-configuration velocity. In this chapter two forms of the generalized inverse, the pseudoinverse and the simpler {1}-inverse, are used to control a general n-axis planar articulated manipulator.

The forces and torques induced at the joints when the tool is in contact with the environment are also examined using a second type of Jacobian matrix called the manipulator Jacobian. With the aid of the manipulator Jacobian, a simple expression for the *compliance* at the end of the arm can be derived. This expression serves as a basis for one of the control methods (impedance control) discussed in Chap. 7.

5-1 THE TOOL-CONFIGURATION JACOBIAN MATRIX

The trajectory-planning problem is formulated naturally in tool-configuration space $\mathbf{R}^6$. However, our control over the motion of the robot tool is indirect, because we can control only the speed or rate at which each joint moves. Recall that the tool configuration specifies the *position* of the tool tip and the *orientation* of the tool. The

relationship between the joint variables $q \in \mathbf{R}^n$ and the tool configuration is specified by the kinematic equations. If we use $x \in \mathbf{R}^6$ to denote the *tool-configuration vector,* then the direct kinematics equations can be formulated in terms of the tool-configuration function w and joint variables q as follows:

$$x = w(q) \tag{5-1-1}$$

Given a desired trajectory $x(t)$ in tool-configuration space, one way to find a corresponding trajectory $q(t)$ in joint space is to invert Eq. (5-1-1) and thereby solve the inverse kinematics problem directly, as was done in Chap. 3. However, finding an explicit closed-form solution to the inverse kinematics problem can be a difficult task, depending upon the kinematic complexity of the manipulator. As an alternative approach, we can examine the *differential relationship* between q and x. If we differentiate both sides of Eq. (5-1-1) with respect to time, this yields:

$$\dot{x} = V(q)\dot{q} \tag{5-1-2}$$

Here $V(q)$ is a $6 \times n$ matrix of partial derivatives of w with respect to q called the *tool-configuration Jacobian matrix,* or simply the *tool Jacobian matrix,* of the manipulator. The component of $V(q)$ appearing in the kth row and jth column is the derivative of $w_k(q)$ with respect to q_j:

$$V_{kj}(q) \triangleq \frac{\partial w_k(q)}{\partial q_j} \qquad 1 \leq k \leq 6,\ 1 \leq j \leq n \tag{5-1-3}$$

We see from Eq. (5-1-2) that for each $q \in \mathbf{R}^n$, the tool Jacobian matrix $V(q)$ is a linear transformation which maps the instantaneous joint-space velocity $\dot{q}$ into the instantaneous tool-configuration velocity $\dot{x}$ as shown in Fig. 5-1.

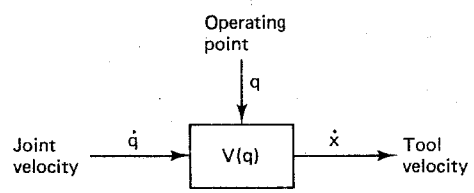


Figure 5-1 The tool configuration Jacobian matrix.

As we shall see, the differential relationship in Eq. (5-1-2) can be used to solve for a joint-space trajectory $q(t)$ given a tool-configuration trajectory $x(t)$. Before we pursue this question, we first pause to examine some examples of tool-configuration Jacobian matrices.

5-1-1 Tool Jacobian Matrix of a Five-Axis Articulated Robot (Rhino XR-3)

Consider the five-axis articulated Rhino XR-3 robot. Recall from Eq. (3-4-1) that the tool-configuration function of this robot is:

$$w(q) = \begin{bmatrix} C_1(a_2C_2 + a_3C_{23} + a_4C_{234} - d_5S_{234}) \\ S_1(a_2C_2 + a_3C_{23} + a_4C_{234} - d_5S_{234}) \\ d_1 - a_2S_2 - a_3S_{23} - a_4S_{234} - d_5C_{234} \\ \hline -[\exp(q_5/\pi)]C_1S_{234} \\ -[\exp(q_5/\pi)]S_1S_{234} \\ -[\exp(q_5/\pi)]C_{234} \end{bmatrix} \tag{5-1-4}$$

The first three components of $w(q)$ represent the tool-tip position, while the last three components of $w(q)$ represent the approach vector of the tool scaled by $\exp(q_5/\pi)$, where q_5 is the tool roll angle. If we differentiate $w(q)$ as indicated in Eq. (5-1-3), this yields the following tool Jacobian matrix, where $v^k(q)$ denotes the kth column of $V(q)$ for $1 \leq k \leq 5$:

$$v^1(q) = \begin{bmatrix} -S_1(a_2C_2 + a_3C_{23} + a_4C_{234} - d_5S_{234}) \\ C_1(a_2C_2 + a_3C_{23} + a_4C_{234} - d_5S_{234}) \\ 0 \\ \hline [\exp(q_5/\pi)]S_1S_{234} \\ -[\exp(q_5/\pi)]C_1S_{234} \\ 0 \end{bmatrix} \tag{5-1-5a}$$

$$v^2(q) = \begin{bmatrix} -C_1(a_2S_2 + a_3S_{23} + a_4S_{234} + d_5C_{234}) \\ -S_1(a_2S_2 + a_3S_{23} + a_4S_{234} + d_5C_{234}) \\ -a_2C_2 - a_3C_{23} - a_4C_{234} + d_5S_{234} \\ \hline -[\exp(q_5/\pi)]C_1C_{234} \\ -[\exp(q_5/\pi)]S_1C_{234} \\ [\exp(q_5/\pi)]S_{234} \end{bmatrix} \tag{5-1-5b}$$

$$v^3(q) = \begin{bmatrix} -C_1(a_3S_{23} + a_4S_{234} + d_5C_{234}) \\ -S_1(a_3S_{23} + a_4S_{234} + d_5C_{234}) \\ -a_3C_{23} - a_4C_{234} + d_5S_{234} \\ \hline -[\exp(q_5/\pi)]C_1C_{234} \\ -[\exp(q_5/\pi)]S_1C_{234} \\ [\exp(q_5/\pi)]S_{234} \end{bmatrix} \tag{5-1-5c}$$

$$v^4(q) = \begin{bmatrix} -C_1(a_4S_{234} + d_5C_{234}) \\ -S_1(a_4S_{234} + d_5C_{234}) \\ -a_4C_{234} + d_5S_{234} \\ \hline -[\exp(q_5/\pi)]C_1C_{234} \\ -[\exp(q_5/\pi)]S_1C_{234} \\ [\exp(q_5/\pi)]S_{234} \end{bmatrix} \tag{5-1-5d}$$

$$v^5(q) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \hline -[\exp\,(q_5/\pi)]C_1\,S_{234}/\pi \\ -[\exp\,(q_5/\pi)]S_1\,S_{234}/\pi \\ -[\exp\,(q_5/\pi)]C_{234}/\pi \end{bmatrix} \tag{5-1-5e}$$

To interpret the tool Jacobian matrix of the Rhino XR-3 robot, it is helpful to examine the distribution of zeros in $V(q)$. Of the thirty entries in $V(q)$, five are zero, as can be seen from the following general expression for $V(q)$, where an X is used to denote a variable element:

$$V(q) = \begin{bmatrix} X & X & X & X & 0 \\ X & X & X & X & 0 \\ 0 & X & X & X & 0 \\ \hline X & X & X & X & X \\ X & X & X & X & X \\ 0 & X & X & X & X \end{bmatrix} \tag{5-1-6}$$

The columns of $V(q)$ correspond to the components of q, while the rows of $V(q)$ correspond to the components of $w(q)$. Consequently, $V_{31}(q) = 0$ can be interpreted to mean that $w_3(q)$ is independent of q_1. That is, the z coordinate of the tool tip cannot be controlled by rotating the arm about its base. Similarly, from $V_{k5}(q) = 0$ for $1 \leq k \leq 3$ we conclude that the tool-tip position p does not depend on the tool roll angle q_5.

5-1-2 Tool Jacobian Matrix of a Four-Axis SCARA Robot (Adept One)

As a second example of a tool-configuration Jacobian matrix, consider a four-axis SCARA robot. Recall from Eq. (3-5-1) that the tool-configuration function of this robot is:

$$w(q) = \begin{bmatrix} a_1 C_1 + a_2 C_{1-2} \\ a_1 S_1 + a_2 S_{1-2} \\ d_1 - q_3 - d_4 \\ \hline 0 \\ 0 \\ -\exp\,(q_4/\pi) \end{bmatrix} \tag{5-1-7}$$

The first three components of $w(q)$ represent the tool-tip position, while the last three components of $w(q)$ represent the approach vector of the tool scaled by $\exp\,(q_4/\pi)$, where q_4 is the tool roll angle. If we differentiate $w(q)$ as indicated in

Eq. (5-1-3), this yields the following tool-configuration Jacobian matrix for the SCARA robot:

$$V(q) = \left[\begin{array}{cccc} -a_1 S_1 - a_2 S_{1-2} & a_2 S_{1-2} & 0 & 0 \\ a_1 C_1 + a_2 C_{1-2} & -a_2 C_{1-2} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -[\exp(q_4/\pi)]/\pi \end{array}\right] \tag{5-1-8}$$

Again, one can draw certain qualitative conclusions based on the distribution of zeros in the tool Jacobian matrix. It is evident from the first two rows of $V(q)$ that the (x, y) coordinates of the tool tip depend only on the first two joint variables $\{q_1, q_2\}$. Similarly, the z coordinate of the tool tip depends only on q_3, while the orientation of the tool is controlled by q_4. The sparse nature of the tool Jacobian matrix reveals that the kinematic structure of a SCARA robot is quite simple.

5-1-3 Tool Jacobian Matrix of a Three-Axis Planar Articulated Robot

As a third example of a tool-configuration Jacobian matrix, consider a three-axis planar articulated robot. Recall from Eq. (3-7-2) that the tool-configuration function of this robot is:

$$w(q) = \left[\begin{array}{c} a_1 C_1 + a_2 C_{12} \\ a_1 S_1 + a_2 S_{12} \\ d_3 \\ \hline 0 \\ 0 \\ \exp(q_3/\pi) \end{array}\right] \tag{5-1-9}$$

The first three components of $w(q)$ represent the tool-tip position, while the last three components of $w(q)$ represent the vector r^3 scaled by $\exp(q_3/\pi)$, where q_3 is the tool roll angle. If we differentiate $w(q)$ as indicated in Eq. (5-1-3), this yields the following tool-configuration Jacobian matrix for the planar articulated robot:

$$V(q) = \left[\begin{array}{ccc} -a_1 S_1 - a_2 S_{12} & -a_2 S_{12} & 0 \\ a_1 C_1 + a_2 C_{12} & a_2 C_{12} & 0 \\ 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & [\exp(q_3/\pi)]/\pi \end{array}\right] \tag{5-1-10}$$

Again, the tool Jacobian matrix is rather sparse. From the third row of $V(q)$ it is evident that the z component of the tool tip is constant. Thus the tool tip moves in a plane orthogonal to the z^0 axis.

5-2 JOINT-SPACE SINGULARITIES

The relationship between the joint-space velocity and the tool-configuration velocity shown in Fig. 5-1 can be used to solve for $\dot{q}$ in terms of $\dot{x}$. This is achieved by multiplying both sides of Eq. (5-1-2) by a generalized inverse of the tool-configuration Jacobian matrix $V(q)$. If $n = 6$ and $V(q)$ is nonsingular, the generalized inverse is simply the standard inverse $V^{-1}(q)$. One of the potential problems with solving for the joint-space velocity in this manner is that at certain points in joint space, the tool Jacobian matrix loses rank; that is, there is a reduction in the number of linearly independent rows and columns. As we approach these points in joint space, $V(q)$ becomes *ill conditioned,* with the numerical solution of Eq. (5-1-2) producing very large joint velocities. The points at which $V(q)$ loses rank are called *joint-space singularities.*

Definition 5-2-1: Joint-Space Singularity. Let $V(q)$ be the $6 \times n$ tool-configuration Jacobian matrix of a robotic arm. Then $\bar{q}$ is a *joint-space singularity* of the arm if and only if:

$$\text{rank } [V(\bar{q})] < \min \{6, n\}$$

Thus a joint-space singularity is any point in joint space at which the tool Jacobian matrix is less than full rank. Sometimes we use the term *workspace singularity.* A point $\bar{w}$ in tool-configuration space is a workspace singularity if and only if $\bar{w} = w(\bar{q})$ for some joint-space singularity $\bar{q}$.

For the most common case, $n \leq 6$, the tool Jacobian matrix is less than full rank if and only if the $n \times n$ matrix $V^T(q)V(q)$ is singular. Since robotic arms are difficult to control when they are operating near joint-space singularities, the following measure of *manipulator dexterity* can be used:

$$\text{dex } (q) \triangleq \det [V^T(q)V(q)] \qquad n \leq 6 \tag{5-2-1}$$

Hence a manipulator is at a joint-space singularity if and only if dex $(q) = 0$. More generally, the dexterity index dex (q) becomes small as q approaches a joint-space singularity. For the redundant case $n > 6$, the determinant of the 6×6 matrix $V(q)V^T(q)$ must be used, and in this case dex (q) has been referred to as the *manipulability* of the robot (Yoshikawa, 1984).

There are two types of joint space singularities. The first type is a *boundary singularity,* which occurs when the tool tip is on the surface of the work envelope, while the second type is an *interior singularity.* The following example illustrates boundary singularities.

Example 5-2-1: Boundary Singularities

Consider the four-axis SCARA robot whose tool-configuration Jacobian matrix is specified in Eq. (5-1-8). From inspection, we see that the last two columns of $V(q)$ are

linearly independent of each other and the first two columns. Thus $V(q)$ loses full rank if and only if the 2×2 submatrix in the upper left corner becomes singular. The determinant of this submatrix is:

$$\begin{aligned}\Delta &= (-a_1 S_1 - a_2 S_{1-2})(-a_2 C_{1-2}) - a_2 S_{1-2}(a_1 C_1 + a_2 C_{1-2}) \\ &= a_1 a_2 S_1 C_{1-2} + a_2^2 S_{1-2} C_{1-2} - a_1 a_2 S_{1-2} C_1 - a_2^2 S_{1-2} C_{1-2} \\ &= a_1 a_2 (S_1 C_{1-2} - C_1 S_{1-2}) \\ &= a_1 a_2 S_2\end{aligned}$$

Consequently, the tool Jacobian matrix of the SCARA robot is less than full rank if and only if $S_2 = 0$. This will occur when the elbow angle q_2 is an integer multiple of π. From Fig. 4-3, we see that when $q_2 = 0$ the arm is reaching straight out with the tool tip on the outer surface of the work envelope, whereas when $|q_2| = \pi$ the arm is folded straight back with the tool tip on the inside surface of the work envelope.

Boundary singularities are not particularly serious, because they can be avoided by simply performing the manipulations sufficiently far from the surface of the work envelope. However, there is another type of singularity that is potentially more troublesome, namely, an interior singularity. Interior singularities occur inside the work envelope when two or more of the axes of the robot form a straight line, that is, become *collinear*. Here the effects of a rotation about one of the axes can be canceled by a counteracting rotation about the other axis. Thus the tool configuration remains constant even though the robot moves in joint space. We refer to these motions of the joints which produce no movement of the tool as *self-motions* of the manipulator. The following example illustrates a locus of interior singularities and the associated self motions of the manipulator.

Example 5-2-2: Interior Singularities

Consider a five-axis articulated robot with tool pitch and tool roll motion. For convenience, we assume that the wrist is spherical, that is $a_4 = 0$. This is the case, for example, with the Microbot Alpha II robot discussed in Chap. 2. Suppose we evaluate the tool-configuration Jacobian matrix along the following locus of points in joint space:

$$q(\beta) = [q_1, -\beta, 2\beta - \pi, -\beta, q_5]^T \qquad 0 < \beta < \frac{\pi}{2}$$

Suppose the upper arm and the forearm are of the same length, $a_3 = a_2$. Using this fact together with $a_4 = 0$, we find from Eq. (5-1-5a) that the first column of the tool Jacobian matrix in this case reduces to:

$$v^1[q(\beta)] = 0 \qquad 0 < \beta < \frac{\pi}{2}$$

Clearly, $V(q)$ loses full rank along the line $q = q(\beta)$ for $0 < \beta < \pi/2$. Therefore $q(\beta)$ represents a *locus* of interior singularities for the articulated robot. This particular locus of points corresponds to the tool being directly above the base and pointing straight up, with the tool roll axis collinear with the base axis as shown in Fig. 5-2.

Notice that if the base axis rotates by γ while the tool roll axis rotates by $-\gamma$, the tool configuration remains fixed even though the robot moves in joint space. This is an example of self-motion of the manipulator. In addition to occurring at interior joint-

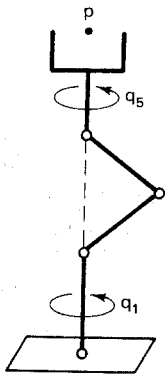


Figure 5-2 An interior joint-space singularity.

space singularities, self-motions are characteristic of redundant robots, that is, robots with more than six axes.

Exercise 5-2-1: Rhino XR-3 Robot Singularities. Find a locus of interior singularities for the Rhino XR-3 robot with $a_4 \neq 0$. *Hint:* Set $p_1 = p_2 = 0$ and $r^3 = i^3$, and solve for q using the inverse kinematics equations.

Excercise 5-2-2: Planar Articulated Robot Singularities. Consider the three-axis planar articulated robot whose tool-configuration Jacobian matrix was developed in Eq. (5-1-10). Show that if $a_2 = a_1$, then $q = [q_1, \pi, q_3]^T$ is a locus of singularities. Which axes are collinear in this case?

Exercise 5-2-3: Human Arm. The human arm is kinematically redundant because it has seven joints, not counting the joints in the articulated fingers, each of which is a small manipulator in its own right. Place the palm of your hand (the tool) flat on the top of a desk. Keep your torso motionless, and see whether you can move your arm. If so, what types of motions are these?

5-3 GENERALIZED INVERSES

The differential relation between joint-space velocity and tool-configuration velocity shown in Fig. 5-1 can be used to solve for $\dot{q}$ in terms of $\dot{x}$. To develop a general solution, one that is valid when the number of axes n is arbitrary, it is useful to first examine the notion of a generalized inverse of a matrix.

Definition 5-3-1: Generalized Inverse. Let A be an $m \times n$ matrix. Then an $n \times m$ matrix X is a *generalized inverse* of A if and only if it satisfies at least property 1 or property 2 of the following list of four properties:

1. $AXA = A$
2. $XAX = X$

3. $(AX)^T = AX$
4. $(XA)^T = XA$

The most well known generalized inverse of A is the *Moore-Penrose inverse*, or *pseudoinverse*, of A, which is denoted by A^+. The pseudoinverse satisfies all four properties in Def. 5-3-1. It is a generalization of the standard inverse, in the sense that A^+ reduces to A^{-1} whenever A is nonsingular. The pseudoinverse always exists and is unique (Ben-Israel and Greville, 1974). Moreover, when the matrix A has full rank, simple explicit expressions for the pseudoinverse can be obtained, as can be seen from the following result:

Proposition 5-3-1: Computing A^+. Let A be an $m \times n$ matrix, and let A^+ be the pseudoinverse of A. If A is of full rank, then A^+ can be computed as follows:

$$A^+ = \begin{cases} A^T[AA^T]^{-1} & m \leq n \\ A^{-1} & m = n \\ [A^TA]^{-1}A^T & m \geq n \end{cases}$$

Proof. Since we have candidates for A^+, it is simply a matter of evaluating them to see whether they satisfy the four properties in Def. 5-3-1. Suppose $m \leq n$. Since A is of full rank, it follows that rank $(A) = m$ and the $m \times m$ matrix AA^T is invertible. Thus:

$$\begin{aligned} AA^+A &= A[A^T(AA^T)^{-1}]A \\ &= (AA^T)(AA^T)^{-1}A \\ &= A \\ A^+AA^+ &= [A^T(AA^T)^{-1}]A[A^T(AA^T)^{-1}] \\ &= [A^T(AA^T)^{-1}](AA^T)(AA^T)^{-1} \\ &= A^T(AA^T)^{-1} \\ &= A^+ \\ (AA^+)^T &= \{A[A^T(AA^T)^{-1}]\}^T \\ &= [(AA^T)(AA^T)^{-1}]^T \\ &= I^T \\ &= I \\ &= (AA^T)(AA^T)^{-1} \\ &= A[A^T(AA^T)^{-1}] \\ &= AA^+ \end{aligned}$$

$$
\begin{aligned}
(A^+A)^T &= \{[A^T(AA^T)^{-1}]A\}^T \\
&= A^T[A^T(AA^T)^{-1}]^T \\
&= A^T[(AA^T)^{-1}]^TA \\
&= A^T[(AA^T)^T]^{-1}A \\
&= A^T(AA^T)^{-1}A \\
&= A^+A
\end{aligned}
$$

Hence, when $m \leq n$ and A is of full rank, $A^+ = A^T[AA^T]^{-1}$. For the case when $m = n$ and A is of full rank, A^{-1} exists and:

$$
\begin{aligned}
A^+ &= A^T[AA^T]^{-1} \\
&= A^T[A^T]^{-1}A^{-1} \\
&= A^{-1}
\end{aligned}
$$

The final case, when $m \geq n$ and A is of full rank, can be verified using steps analogous to those used for the first case. This is left as an exercise.

Example 5-3-1: Pseudoinverse

Let A be the following 2×3 matrix:

$$
A = \begin{bmatrix} 1 & 0 & 2 \\ 1 & -1 & 0 \end{bmatrix}
$$

It is clear from inspection that rank $(A) = 2$. Thus, applying Prop. 5-3-1, we have:

$$
\begin{aligned}
A^+ &= A^T[AA^T]^{-1} \\
&= \begin{bmatrix} 1 & 1 \\ 0 & -1 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} 5 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \\
&= \frac{1}{9} \begin{bmatrix} 1 & 1 \\ 0 & -1 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ -1 & 5 \end{bmatrix} \\
&= \frac{1}{9} \begin{bmatrix} 1 & 4 \\ 1 & -5 \\ 4 & -2 \end{bmatrix}
\end{aligned}
$$

In view of Prop. 5-3-1, simple explicit expressions for the pseudoinverse of a matrix are available when the matrix is of full rank. Recall that the tool-configuration Jacobian matrix $V(q)$ is of full rank as long as q is not a joint-space singularity.

Exercise 5-3-1: Pseudoinverse. Verify that the matrix A^+ found in Example 5-3-1 satisfies the four properties listed in Def. 5-3-1.

The importance of the pseudoinverse lies in its applicability to solving systems of linear algebraic equations. Consider a system of the following general form where A is an $m \times n$ real matrix and $b \in \mathbf{R}^m$:

$$Ax = b \tag{5-3-1}$$

If A is of full rank, then Eq. (5-3-1) will have a solution x whenever $m \leq n$, although the solution may not be unique. If $m > n$, then there are more constraints than unknown components of x, and a solution may or may not exist. In this case, we instead attempt to find an x which is as *close* to a solution as possible in the sense of minimizing the norm of the *residual vector*, $r(x) = Ax - b$. If the Euclidean norm of $r(x)$ is used, then x is referred to as a *least-squares solution* of Eq. (5-3-1). The pseudoinverse of A produces a special least-squares solution, as can be seen from the following result:

Proposition 5-3-2: Pseudoinverse. Let A^+ be the pseudoinverse of the $m \times n$ matrix A in Eq. (5-3-1). Among all least-squares solutions of Eq. (5-3-1), the least-squares solution x^0 whose norm is smallest is:

$$x^0 = A^+ b$$

The interested reader can refer to Ben-Israel and Greville (1974) for a proof of Prop. 5-3-2. Proposition 5-3-2 can be summarized in words by saying that the pseudoinverse provides the minimum least-squares solution of Eq. (5-3-1). Since a least-squares solution minimizes $\|Ax - b\|$, a least-squares solution is an actual solution whenever a solution to $Ax = b$ exists.

Example 5-3-2: Linear Algebraic System

Consider the following linear algebraic system of two equations in three unknowns:

$$\begin{bmatrix} 1 & 0 & 2 \\ 1 & -1 & 0 \end{bmatrix} x = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$$

Since $m < n$ and A has full rank, this system has many solutions. Using Prop. 5-3-2 and the results of Example 5-3-1, the smallest solution in the sense of minimizing $\|x\|$ is:

$$\begin{aligned} x &= A^+ b \\ &= \frac{1}{9} \begin{bmatrix} 1 & 4 \\ 1 & -5 \\ 4 & -2 \end{bmatrix} \begin{bmatrix} 3 \\ -2 \end{bmatrix} \\ &= \frac{1}{9} [-5, 13, 16]^T \end{aligned}$$

In this case, $\|x\| = 2.357$, and for all other solutions of this system, $\|x\| \geq 2.357$.

5-4 RESOLVED-MOTION RATE CONTROL: $n \leq 6$

The differential relation between joint-space velocity and tool-configuration velocity in Eq. (5-1-2) can now be used to solve for $\dot{q}$ in terms of $\dot{x}$. First we examine the most common case, when the number of axes is less than or equal to 6.

Proposition 5-4-1: Resolved-Motion Rate Control: $n \leq 6$. Let $x(t)$ be a differentiable tool-configuration trajectory which lies inside the work envelope and which does not go through any workspace singularities, and let $V(q)$ be the $6 \times n$ tool-configuration Jacobian matrix where $n \leq 6$. Then a joint-space trajectory $q(t)$ corresponding to $x(t)$ can be obtained by solving the following nonlinear differential equation:

$$\dot{q} = [V^T(q)V(q)]^{-1}V^T(q)\dot{x} \qquad q(0) = w^{-1}(x(0))$$

Proof. Since $x(t)$ lies inside the work envelope, there is at least one $q(t)$ such that $w(q(t)) = x(t)$, where w is the tool-configuration function. Therefore the least-squares solution of Eq. (5-1-2) is a solution. Consequently, from Prop. 5-3-2, a solution for $\dot{q}$ can be obtained from Eq. (5-1-2) by multiplying both sides by the pseudoinverse of $V(q)$. That is,

$$\dot{q} = V^+(q)\dot{x}$$

Since $x(t)$ does not go through any workspace singularities, $V(q)$ has full rank for all $q(t)$ corresponding to $x(t)$. But $V(q)$ is $6 \times n$, with $n \leq 6$. Therefore rank $[V(q)] = n$, and from Prop. 5-3-1, the pseudoinverse of $V(q)$ is given by:

$$V^+(q) = [V^T(q)V(q)]^{-1}V^T(q)$$

The initial condition supplied with the nonlinear differential equation in Prop. 5-4-1 ensures that the joint-space trajectory $q(t)$ and the tool-configuration trajectory $x(t)$ correspond to the same physical point at time $t = 0$. Here the notation w^{-1} refers to the inverse of the tool-configuration function. Therefore $q(0)$ can be computed from $x(0)$ using the inverse kinematic equations.

If the robot being controlled is general in the sense of having exactly six axes, then $V(q)$ is a square matrix, in which case $V^+(q)$ reduces to $V^{-1}(q)$. Consequently, when $n = 6$, the rate-control equations in Prop. 5-4-1 simplify to:

$$\dot{q} = V^{-1}(q)\dot{x} \qquad q(0) = w^{-1}(x(0)) \tag{5-4-1}$$

The control technique described in Prop. 5-4-1, and more specifically in Eq. (5-4-1), was introduced by Whitney (1969) and is commonly referred to as *resolved-motion rate control*. Here the motion of the robot in tool-configuration space is *resolved* into its joint-space components through the tool-configuration Jacobian matrix.

Example 5-4-1: SCARA Robot

Consider the application of resolved-motion rate control to a four-axis SCARA robot. First we must compute the pseudoinverse of $V(q)$. To simplify the notation, we use $\{\alpha, \beta, \gamma, \ldots\}$ for the nonconstant components of $V(q)$. From Eq. (5-1-8), we then have:

$$V = \begin{bmatrix} \alpha & \beta & 0 & 0 \\ \gamma & \delta & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \epsilon \end{bmatrix}$$

$$V^T = \begin{bmatrix} \alpha & \gamma & 0 & 0 & 0 & 0 \\ \beta & \delta & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \epsilon \end{bmatrix}$$

Computing V^TV, we get the following 4×4 block diagonal matrix:

$$V^TV = \left[\begin{array}{cc:cc} \alpha^2 + \gamma^2 & \alpha\beta + \gamma\delta & 0 & 0 \\ \alpha\beta + \gamma\delta & \beta^2 + \delta^2 & 0 & 0 \\ \hdashline 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \epsilon^2 \end{array}\right]$$

Since V^TV is a block diagonal matrix, the inverse is also block diagonal. Let Δ denote the determinant of the upper left diagonal block. The inverse of V^TV can be expressed in terms of Δ as:

$$[V^TV]^{-1} = \frac{1}{\Delta}\left[\begin{array}{cc:cc} \beta^2 + \delta^2 & -(\alpha\beta + \gamma\delta) & 0 & 0 \\ -(\alpha\beta + \gamma\delta) & \alpha^2 + \gamma^2 & 0 & 0 \\ \hdashline 0 & 0 & \Delta & 0 \\ 0 & 0 & 0 & \Delta/\epsilon^2 \end{array}\right]$$

$$\Delta = (\alpha^2 + \gamma^2)(\beta^2 + \delta^2) - (\alpha\beta + \gamma\delta)^2$$

Since $\epsilon = -[\exp(q_4/\pi)]/\pi \neq 0$, it follows that $[V^TV]^{-1}$ is well defined as long as $\Delta \neq 0$. This will be the case if $x(t)$ does not go through a workspace singularity. It remains to multiply $[V^TV]^{-1}$ by V^T to compute V^+. This yields:

$$V^+ = \frac{1}{\Delta}\begin{bmatrix} (\alpha\delta - \beta\gamma)\delta & (\beta\gamma - \alpha\delta)\beta & 0 & 0 & 0 & 0 \\ (\beta\gamma - \alpha\delta)\gamma & (\alpha\delta - \beta\gamma)\alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & -\Delta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \Delta/\epsilon \end{bmatrix}$$

From the expression for V^+, we can write down the rate-control equations using Prop. 5-4-1 as follows:

$$\dot{q}_1 = \frac{(\alpha\delta - \beta\gamma)\delta\dot{x}_1 + (\beta\gamma - \alpha\delta)\beta\dot{x}_2}{\Delta}$$

$$\dot{q}_2 = \frac{(\beta\gamma - \alpha\delta)\gamma\dot{x}_1 + (\alpha\delta - \beta\gamma)\alpha\dot{x}_2}{\Delta}$$

$$\dot{q}_3 = -\dot{x}_3$$

$$\dot{q}_4 = \frac{\dot{x}_6}{\epsilon}$$

Note that for $x(t)$ to be a trajectory in the robot work envelope we must have $x_4(t) = 0$ and $x_5(t) = 0$, because the approach vector of a SCARA robot is always vertical. As the trajectory of the arm approaches a singularity, Δ becomes small, and the joint velocities $\dot{q}_1$ and $\dot{q}_2$ become large. This is precisely the ill-conditioned behavior that is to be avoided through proper trajectory planning.

5-5 RATE CONTROL OF REDUNDANT ROBOTS: $n > 6$

One way to get around the breakdown of resolved-motion rate control near joint-space singularities is to add more joints to the robotic arm. If the number of joints exceeds the dimension of the tool-configuration space, then the robotic arm is said to be *kinematically redundant*. It follows that any arm with more than six joints is always kinematically redundant. The human arm is a familiar case in point. The extra degrees of freedom represented by the redundant joints can be used to avoid singularities as well as accomplish other objectives such as reaching around or between obstacles. Since the joint velocities computed through resolved-motion rate control tend to be large near singularities, one way to *implicitly* avoid singularities is to find a solution to Eq. (5-1-2) which minimizes $\|\dot{q}\|$. The pseudoinverse of the tool-configuration Jacobian matrix yields precisely such a solution, as can be seen from the following result:

Proposition 5-5-1: Rate Control of Redundant Robots: $n > 6$. Let $x(t)$ be a differentiable tool-configuration trajectory which lies inside the work envelope and which does not go through any workspace singularities, and let $V(q)$ be the $6 \times n$ tool-configuration Jacobian matrix, where $n > 6$. Then a joint-space trajectory $q(t)$ corresponding to $x(t)$ which minimizes $\|\dot{q}(t)\|$ can be obtained by solving the following nonlinear differential equation:

$$\dot{q} = V^T(q)[V(q)V^T(q)]^{-1}\dot{x} \qquad q(0) = w^{-1}(x(0))$$

Proof. Since $x(t)$ lies inside the work envelope, there is at least one $q(t)$ such that $w(q(t)) = x(t)$, where w is the tool-configuration function. Consequently, from Prop. 5-3-2, a solution for $\dot{q}$ can be obtained from Eq. (5-1-2) by multiplying both sides by the pseudoinverse of $V(q)$. Thus:

$$\dot{q} = V^+(q)\dot{x}$$

Since $x(t)$ does not go through any workspace singularities, $V(q)$ has full rank for all $q(t)$ corresponding to $x(t)$. But $V(q)$ is $6 \times n$ with $n > 6$. Therefore rank $[V(q)] = 6$, and from Prop. 5-3-1, the pseudoinverse of $V(q)$ is given by:

$$V^+(q) = V^T(q)[V(q)V^T(q)]^{-1}$$

Finally, from Prop. 5-3-2, the pseudoinverse solution of Eq. (5-1-2) is the solution whose norm is smallest.

Note that $\|\dot{q}(t)\|$ can be thought of as the overall *speed* of the arm at time t. Thus, for a redundant robot, the pseudoinverse of the tool-configuration Jacobian

matrix produces the minimum-speed joint-space trajectory $q(t)$ corresponding to the tool-configuration trajectory $x(t)$.

Proposition 5-5-1 can be generalized in a simple but important way. The solution of Eq. (5-1-2) specified in Prop. 5-5-1 is *optimal*, in the sense of minimizing $f(\dot{q}) = \dot{q}^T\dot{q} = \|\dot{q}\|^2$ subject to the constraint $\dot{x} = V(q)\dot{q}$. Consider instead the following generalization of the performance index:

$$f(\dot{q}) \triangleq \frac{\dot{q}^T W \dot{q}}{2} \tag{5-5-1}$$

Here W is a symmetric positive-definite *weighting matrix*. For example, W might be diagonal with positive diagonal elements. To find a $\dot{q}$ which minimizes $f(\dot{q})$ while satisfying $\dot{x} = V(q)\dot{q}$, we use the method of Lagrange multipliers (Athans and Falb, 1966). That is, let $\lambda \in \mathbf{R}^6$, and consider the following augmented performance index, which depends on both $\dot{q}$ and λ:

$$F(\dot{q}, \lambda) \triangleq f(\dot{q}) + \lambda^T(\dot{x} - V(q)\dot{q}) \tag{5-5-2}$$

If we minimize $F(\dot{q}, \lambda)$ with respect to $\dot{q}$ and λ, this is equivalent to minimizing $f(\dot{q})$ with respect to $\dot{q}$ subject to the constraint $\dot{x} = V(q)\dot{q}$. That is, through the introduction of the *Lagrange multiplier* vector λ we have converted a constrained minimization of dimension n into an equivalent unconstrained minimization of dimension $n + 6$. If the pair $(\dot{q}, \lambda)$ is to minimize $F(\dot{q}, \lambda)$, then the derivatives of $F(\dot{q}, \lambda)$ with respect to $\dot{q}$ and λ must be zero. From Eqs. (5-5-1) and (5-5-2), taking the derivatives of $F(\dot{q}, \lambda)$ with respect to $\dot{q}$ and λ and setting them to zero yields:

$$W\dot{q} - V^T(q)\lambda = 0 \tag{5-5-3}$$

$$\dot{x} - V(q)\dot{q} = 0 \tag{5-5-4}$$

Recall that W is positive-definite and therefore nonsingular. Multiplying both sides of Eq. (5-5-3) by W^{-1}, we solve for $\dot{q}$ in terms of λ as follows:

$$\dot{q} = W^{-1}V^T(q)\lambda \tag{5-5-5}$$

If the expression for $\dot{q}$ in Eq. (5-5-5) is substituted in Eq. (5-5-4), this generates a 6×6 matrix coefficient of λ. Since $n > 6$, this matrix coefficient, $V(q)W^{-1}V^T(q)$, is nonsingular as long as q is not a joint-space singularity. Hence we can combine Eqs. (5-5-4) and (5-5-5) and solve for the Lagrange multiplier vector λ as follows:

$$\lambda = [V(q)W^{-1}V^T(q)]^{-1}\dot{x} \tag{5-5-6}$$

Finally, the expression for λ can be substituted in Eq. (5-5-5) to generate the following nonlinear differential equation for the joint rates of a kinematically redundant manipulator:

$$\dot{q} = W^{-1}V^T(q)[V(q)W^{-1}V^T(q)]^{-1}\dot{x} \qquad q(0) = w^{-1}(x(0)) \tag{5-5-7}$$

Note that Eq. (5-5-7) reduces to the formulation in Prop. 5-5-1 when *uniform weighting*, $W = I$, is used. The utility of Eq. (5-5-7) is that joints near the base of the robot, which typically require large actuators, can be given more weight in the

specification of W. Therefore the solution which minimizes $f(\dot{q})$ in Eq. (5-5-1) will be less likely to require rapid movements from these large actuators.

Robots with more than six axes are always kinematically redundant. However, it is also possible for robots with fewer axes to be redundant if the dimension of the tool-configuration space is restricted. Consider, for example, the *planar articulated robot* shown in Fig. 5-3. Here the *length* of link k is a_k and the *local angle* that link k makes with respect to link $k-1$ is θ_k for $1 \leq k \leq n$. Since the robot is operating in a plane orthogonal to the z^0 axis, there are only two positional degrees of freedom and one orientational degree of freedom. Thus the tool-configuration space for a planar robot is three-dimensional. Suppose we define the tool-configuration vector as follows:

$$x = \begin{bmatrix} p_1 \\ p_2 \\ \psi \end{bmatrix} \tag{5-5-8}$$

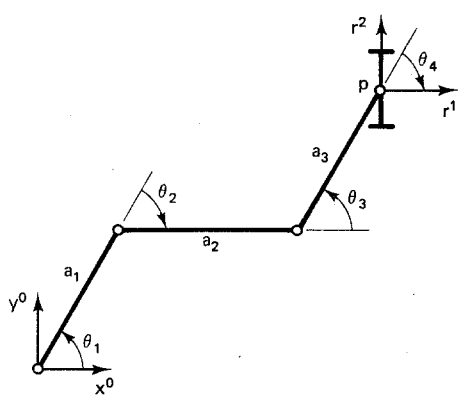


Figure 5-3 A redundant planar articulated arm.

Here the first two components of x represent the *wrist position,* while the last component of x represents the *global tool angle* ψ, which is the angle the normal vector r^1 makes relative to the x^0 axis. Since the tool-configuration space is three-dimensional, it follows that an n-axis planar articulated manipulator is kinematically redundant whenever $n > 3$. The direct kinematic equations of a planar manipulator can be simplified if we use the *global joint angles* measured relative to the x^0 axis. That is, let:

$$q_k \triangleq \sum_{j=1}^{k} \theta_j \qquad 1 \leq k \leq n \tag{5-5-9}$$

Here the global angle q_k is simply the sum of the first k local angles $\{\theta_1, \theta_2, \ldots, \theta_k\}$. The local angles can be easily recovered from the global angles, yet the kinematic equations simplify considerably when the global angles are used. From Fig. 5-3 it is clear that the tool-configuration function w can be expressed in terms of the global joint angles as follows:

$$w_1(q) = \sum_{k=1}^{n-1} a_k C_k \tag{5-5-10a}$$

$$w_2(q) = \sum_{k=1}^{n-1} a_k S_k \tag{5-5-10b}$$

$$w_3(q) = q_n \tag{5-5-10c}$$

To apply resolved-motion rate control to the n-axis planar articulated arm, we must first compute the tool-configuration Jacobian matrix. Differentiating Eq. (5-5-10), we have:

$$V(q) = \left[\begin{array}{cccc|c} -a_1 S_1 & -a_2 S_2 & \cdots & -a_{n-1} S_{n-1} & 0 \\ a_1 C_1 & a_2 C_2 & \cdots & a_{n-1} C_{n-1} & 0 \\ \hline 0 & 0 & \cdots & 0 & 1 \end{array}\right] \tag{5-5-11}$$

Notice that the tool Jacobian matrix is a $3 \times n$ block diagonal matrix. The block diagonal structure is a consequence of the fact that the kinematic equations have been partitioned at the wrist.

Exercise 5-5-1: Local Angles. The local angles θ in Eq. (5-5-9) can be reconstructed from the global angles q using $\theta = Hq$. Find the $n \times n$ reconstruction matrix H. Find H^{-1}.

Example 5-5-1: Planar Articulated Robot

Suppose we use the expression for $V(q)$ in Eq. (5-5-11) to apply resolved-motion rate control to the general planar articulated arm in Fig. 5-3. First we must compute the pseudoinverse of the tool Jacobian matrix. From Eq. (5-5-11) and Prop. 5-3-1, we have:

$$VV^T = \left[\begin{array}{cc|c} \sum_{k=1}^{n-1} a_k^2 S_k^2 & -\sum_{k=1}^{n-1} a_k^2 S_k C_k & 0 \\ -\sum_{k=1}^{n-1} a_k^2 S_k C_k & \sum_{k=1}^{n-1} a_k^2 C_k^2 & 0 \\ \hline 0 & 0 & 1 \end{array}\right]$$

Since VV^T is block diagonal, so is its inverse. The inverse will exist if joint-space singularities, which correspond to the arm being straight out, are avoided. In this case, the inverse is:

$$[VV^T]^{-1} = \frac{1}{\Delta}\left[\begin{array}{cc|c} \sum_{k=1}^{n-1} a_k^2 C_k^2 & \sum_{k=1}^{n-1} a_k^2 S_k C_k & 0 \\ \sum_{k=1}^{n-1} a_k^2 S_k C_k & \sum_{k=1}^{n-1} a_k^2 S_k^2 & 0 \\ \hline 0 & 0 & \Delta \end{array}\right]$$

Here Δ denotes the determinant of the diagonal block in the upper left corner of VV^T. That is:

$$\Delta = \left(\sum_{k=1}^{n-1} a_k^2 S_k^2\right)\left(\sum_{j=1}^{n-1} a_j^2 C_j^2\right) - \left(\sum_{k=1}^{n-1} a_k^2 S_k C_k\right)\left(\sum_{j=1}^{n} a_j^2 S_j C_j\right)$$

$$= \sum_{k=1}^{n-1}\sum_{j=1}^{n-1} a_k^2 a_j^2 S_k^2 C_j^2 - \sum_{k=1}^{n-1}\sum_{j=1}^{n-1} a_k^2 a_j^2 S_k C_k S_j C_j$$

$$= \sum_{k=1}^{n-1}\sum_{j=1}^{n-1} a_k^2 a_j^2 (S_k^2 C_j^2 - S_k C_k S_j C_j)$$

$$= \sum_{k=1}^{n-1}\sum_{j=1}^{n-1} a_k^2 a_j^2 S_k C_j (S_k C_j - C_k S_j)$$

$$= \sum_{k=1}^{n-1}\sum_{j=1}^{n-1} a_k^2 a_j^2 S_k C_j S_{k-j}$$

Note that $\Delta = 0$ if $S_{k-j} = 0$ for all $j \neq k$. That is, the arm is at a joint-space singularity if $q_k - q_j$ is an integer multiple of π for every $j \neq k$. This corresponds to all of the links of the arm being aligned. If $\Delta \neq 0$, then the pseudoinverse can be obtained from Prop. 5-3-1 by premultiplying by V^T. After simplification, the result is:

$$V^+ = \frac{1}{\Delta}\left[\begin{array}{cc|c} a_1 \sum_{k=1}^{n-1} a_k^2 C_k S_{k-1} & a_1 \sum_{k=1}^{n-1} a_k^2 S_k S_{k-1} & 0 \\ a_2 \sum_{k=1}^{n-1} a_k^2 C_k S_{k-2} & a_2 \sum_{k=1}^{n-1} a_k^2 S_k S_{k-2} & 0 \\ \vdots & \vdots & \vdots \\ a_{n-1} \sum_{k=1}^{n-1} a_k^2 C_k S_{k-n+1} & a_{n-1} \sum_{k=1}^{n-1} a_k^2 S_k S_{k-n+1} & 0 \\ \hline 0 & 0 & \Delta \end{array}\right]$$

Finally, the minimum joint speed rate-control equations for the n-axis planar articulated robot corresponding to a differentiable tool-configuration trajectory $x(t)$ are:

$$\dot{q}_1 = \frac{\left(a_1 \sum_{k=1}^{n-1} a_k^2 C_k S_{k-1}\right)\dot{x}_1 + \left(a_1 \sum_{k=1}^{n-1} a_k^2 S_k S_{k-1}\right)\dot{x}_2}{\Delta}$$

$$\dot{q}_2 = \frac{\left(a_2 \sum_{k=1}^{n-1} a_k^2 C_k S_{k-2}\right)\dot{x}_1 + \left(a_2 \sum_{k=1}^{n-1} a_k^2 S_k S_{k-2}\right)\dot{x}_2}{\Delta}$$

$$\vdots$$

$$\dot{q}_{n-1} = \frac{\left(a_{n-1} \sum_{k=1}^{n-1} a_k^2 C_k S_{k-n+1}\right)\dot{x}_1 + \left(a_{n-1} \sum_{k=1}^{n-1} a_k^2 S_k S_{k-n+1}\right)\dot{x}_2}{\Delta}$$

$$\dot{q}_n = \dot{x}_3$$

As the tool-configuration trajectory $x(t)$ approaches a joint-space singularity, the determinant Δ approaches zero. In this case all of the joint velocities except the last one become large. Note that $S_{k-j} = \sin(q_k - q_j) \neq \sin(q_{k-j})$.

5-6 RATE CONTROL USING {1}-INVERSES

The rate-control equations based on the pseudoinverse of the tool-configuration Jacobian matrix are *optimal,* in the sense of minimizing the overall joint speed $\|\dot{q}\|$. However, for kinematically redundant manipulators there are also other objectives we might want to consider. For example, if the robot is operating in a workspace cluttered with obstacles, then we may want to exploit the extra degrees of freedom to reach around or between obstacles and thereby manipulate otherwise inaccessible objects. To examine this problem further, consider again a linear algebraic system of equations where A is an $m \times n$ real matrix and $b \in \mathbf{R}^m$:

$$Ax = b \tag{5-6-1}$$

Suppose $m < n$ and A is of full rank, which means that multiple solutions exist. If something other than the minimum norm solution is to be computed, then there is no particular advantage to using the computationally expensive pseudoinverse of A. Instead, a simpler generalized inverse called a {1}-inverse can be employed. A *{1}-inverse* of A is denoted $A^{(1)}$ and is defined as any generalized inverse of A satisfying property 1 of the four properties listed in Def. 5-3-1. The Moore-Penrose inverse, or pseudoinverse, is the most well known {1}-inverse, but in addition to satisfying property 1, it must also satisfy properties 2 through 4. Consequently, the pseudoinverse is more difficult to compute than a {1}-inverse. Indeed, if the $m \times n$ matrix A is of full rank and the first m columns of A are linearly independent, then it is a simple matter to compute $A^{(1)}$, as can be seen from the following result:

Proposition 5-6-1: Computing a {1}-Inverse. Let $A = [B, C]$ be an $m \times n$ matrix with $m < n$. If the $m \times m$ submatrix B is nonsingular, then the following matrix is a {1}-inverse of A:

$$A^{(1)} = \begin{bmatrix} B^{-1} \\ \hline 0 \end{bmatrix}$$

Proposition 5-6-1 can be verified directly by showing that $AA^{(1)}A = A$, which is property 1 of Def. 5-3-1. If A is of full rank and $m < n$, then there is no loss of generality in assuming that the first m columns are linearly independent, because the columns of A can always be reordered, if needed. For the linear algebraic system in Eq. (5-6-1), reordering the columns of A is equivalent to reindexing the components of the vector x.

Exercise 5-6-1: Block Diagonal Matrices. Suppose A is a block diagonal matrix with diagonal blocks B and C. That is, $A = \text{diag}\{B, C\}$, where B and C are not necessarily square. Show that $A^{(1)}$ is also block diagonal with:

$$A^{(1)} = \text{diag}\{B^{(1)}, C^{(1)}\}$$

Exercise 5-6-2: Inverse of Transpose. Show that the {1}-inverse operation

and the matrix transpose operation commute. That is, show that:

$$[A^T]^{(1)} = [A^{(1)}]^T$$

In view of Prop. 5-6-1, computing a {1}-inverse of a tool Jacobian matrix is simpler than computing the pseudoinverse, because two matrix multiplication operations and two transpose operations are avoided. Perhaps more important, the final expression for $V^{(1)}$ tends to be simpler than the final expression for V^+, as can be seen from the following example:

Example 5-6-1: Planar Articulated Arm

Consider again the planar articulated robot shown in Fig. 5-3. The tool-configuration Jacobian matrix has been previously computed in Eq. (5-5-4). Since $V(q)$ is block diagonal, it follows from Exercise 5-6-1 that $V^{(1)}(q)$ is also block diagonal. Assuming that the first two columns of $V(q)$ are linearly independent, a {1}-inverse of $V(q)$ can be computed from Prop. 5-6-1 as follows:

$$V^{(1)}(q) = \frac{1}{\Delta}\left[\begin{array}{cc:c} a_2C_2 & a_2S_2 & 0 \\ -a_1C_1 & -a_1S_1 & 0 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ \hdashline 0 & 0 & \Delta \end{array}\right]$$

Here Δ is the determinant of the 2×2 submatrix in the upper left corner of $V(q)$. That is,

$$\begin{aligned} \Delta &= -a_1S_1a_2C_2 + a_1C_1a_2S_2 \\ &= -a_1a_2(S_1C_2 - C_1S_2) \\ &= -a_1a_2S_{1-2} \\ &= a_1a_2S_{2-1} \end{aligned}$$

Thus $V^{(1)}$ is well defined as long as $S_{2-1} \neq 0$. But $S_{2-1} = \sin(q_2 - q_1)$, where $q_2 - q_1 = \theta_2$. Therefore, the formulation of $V^{(1)}$ is valid as long as the first two links in Fig. 5-3 are not aligned. This will be the case if the robot operates strictly in a *left-handed* mode ($0 < \theta_2 < \pi$) or in a *right-handed* mode ($-\pi < \theta_2 < 0$). It is evident by inspection that the final expression for $V^{(1)}(q)$ is considerably simpler than the equivalent expression for $V^+(q)$ computed in Example 5-5-1.

Although a {1}-inverse does not have all of the useful characteristics of the pseudoinverse, it does satisfy one important property: it produces a least-squares solution of Eq. (5-6-1) even though it is not necessarily the *smallest* least-squares solution (Lovass-Nagy et al., 1978). Now if rank $(A) = m$, then Eq. (5-6-1) has a solution for every $b \in \mathbf{R}^m$. This means that the least-squares solution, $x = A^{(1)}b$, is an actual solution when rank $(A) = m$. To generate all possible solutions of Eq. (5-6-1), we add to this particular solution vectors that belong to the null space of A,

that is, solutions of $Ax = 0$. The following is then a *general solution* to Eq. (5-6-1), where the vector $h \in \mathbf{R}^n$ is arbitrary (Lovass-Nagy et al., 1978):

$$x = A^{(1)}b + (I - A^{(1)}A)h \tag{5-6-2}$$

Here the second term in Eq. (5-6-2) is a vector in the null space of A, as can be verified by premultiplying by A and using property 1 of Def. 5-3-1. When rank $(A) = m$, the expression for x in Eq. (5-6-2) is a solution of Eq. (5-6-1) for *every* vector $h \in \mathbf{R}^n$. The challenge, then, is to choose h in such a way as to minimize some desired performance index.

In the case of a redundant n-axis planar articulated robot, the general {1}-inverse formulation of the rate equations, based on Eq. (5-6-2), allows for *direct control* of the rates of the $n - 3$ redundant joints, as can be seen from the following result:

Proposition 5-6-2: Planar Articulated Robot. Let $x(t)$ be a differentiable tool-configuration trajectory that lies inside the work envelope of the planar articulated robot in Fig. 5-3 when it is operating in either a left-handed or a right-handed mode. If $n > 3$, then a joint-space trajectory $q(t)$ corresponding to $x(t)$ can be obtained by solving the following nonlinear differential equation, where the redundant joint rates $r \in \mathbf{R}^{n-3}$ are arbitrary and $q(0) = w^{-1}(x(0))$:

$$\dot{q}_1 = \frac{C_2\dot{x}_1 + S_2\dot{x}_2 + \sum_{k=3}^{n-1} a_k S_{k-2} r_{k-2}}{a_1 S_{2-1}}$$

$$\dot{q}_2 = -\frac{C_1\dot{x}_1 + S_1\dot{x}_2 - \sum_{k=3}^{n-1} a_k S_{k-1} r_{k-2}}{a_2 S_{2-1}}$$

$$\dot{q}_k = r_{k-2} \qquad 3 \leq k \leq n-1$$

$$\dot{q}_n = \dot{x}_3$$

The interested reader is referred to Lovass-Nagy and Schilling (1987) for the derivation of the equations in Prop. 5-6-2. If we compare the rate equations in Prop. 5-6-2 with the pseudoinverse formulation in Example 5-5-1, we see that the {1}-inverse formulation is indeed simpler. However, the utility of the {1}-inverse rate equations lies not so much in a reduction in the number of computations needed for implementation. Rather, the key feature of Prop. 5-6-2 is that the control equations have been *decoupled* into redundant and nonredundant parts where the $n - 3$ redundant joint rates can be specified *arbitrarily*. Notice that the {1}-inverse control scheme dictates that the first two joints control the tool-tip position, while the last joint controls the tool orientation. The remaining $n - 3$ joints in the middle of the arm are *redundant joints* whose rates can be specified arbitrarily subject to the limits of the joint actuators. For example, if $r = 0$, then the $n - 3$ joints in the middle of the arm are frozen in their current position. In this *stiff* configuration, the arm is effectively operating as if it were not redundant. Alternatively, the redundant rates r can be selected so as to avoid collisions with obstacles in the workspace (Lovass-

Nagy and Schilling, 1987). It is also possible to reformulate the {1}-inverse of $V(q)$ in such a way that joints other than the first two are used to control the tool-tip position.

Exercise 5-6-3: Minimum Norm Solution. Show that the general {1}-inverse solution of Eq. (5-6-1) depicted in Eq. (5-6-2) includes the minimum norm, or pseudoinverse, solution as a special case. *Hint:* Try $h = A^{+}b$.

5-7 THE MANIPULATOR JACOBIAN

At this point we turn our attention from differential-motion rate-control methods to a separate but related problem, the analysis of a manipulator in static equilibrium. If a robotic manipulator is to perform meaningful work, the tool must come into contact with the environment. The contact with the environment produces a reaction force and moment at the tool tip, as shown in Fig. 5-4. In applications (for example, paint spraying) where no direct contact occurs, an equivalent force and moment is generated at the tool tip due to the load carried by the robot.

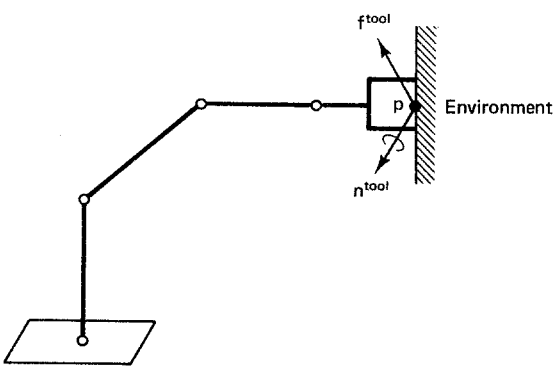


Figure 5-4 Tool in contact with environment.

The external force and moment must be compensated for by internal joint torques and forces if the manipulator is to remain in static equilibrium. To develop a concise relationship between the external forces and moments applied at the tool tip and the internal joint torques and forces they induce, we reexamine the concept of a Jacobian matrix.

Recall that the tool-configuration Jacobian matrix was based on a representation of tool orientation that used a scaled version of the approach vector. In this section we investigate a more traditional formulation of the Jacobian matrix, called the *manipulator Jacobian,* that is based on the notion of infinitesimal translations and rotations of the tool (Whitney, 1972). In particular, let $dp \in \mathbf{R}^3$ represent an *infinitesimal translation* of the tool tip expressed in base coordinates. Similarly, let $d\phi \in \mathbf{R}^3$ represent an *infinitesimal rotation* of the tool also expressed in base coordinates as shown in Fig. 5-5.

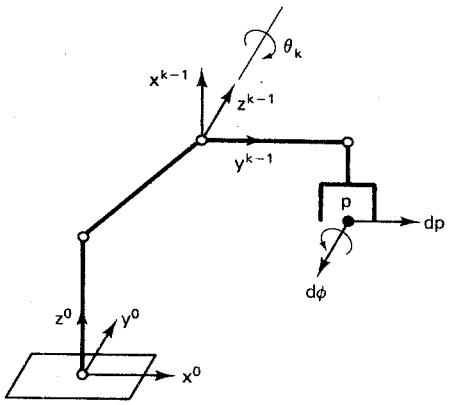


Figure 5-5 Infinitesimal translation and rotation of tool.

We combine the infinitesimal translation dp and the infinitesimal rotation $d\phi$ into a six-dimensional vector du called the *infinitesimal displacement* vector:

$$du \triangleq \begin{bmatrix} dp \\ d\phi \end{bmatrix} \tag{5-7-1}$$

Thus du represents both linear displacement and angular displacement of the tool. The infinitesimal tool displacement is directly related to the infinitesimal joint displacement dq. More specifically, for each point q in joint space $\mathbf{R}^n$, there exists a $6 \times n$ matrix $J(q)$ such that:

$$du = J(q)\, dq \tag{5-7-2}$$

We call $J(q)$ the *manipulator Jacobian matrix* in order to distinguish it from the tool-configuration Jacobian matrix $V(q)$ introduced earlier. Refer to Fig. 5-6 for a pictorial representation of the manipulator Jacobian matrix.

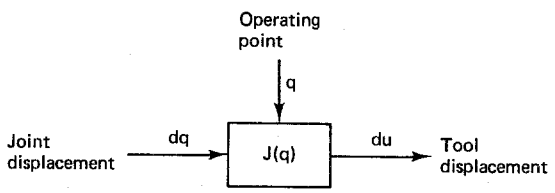


Figure 5-6 The manipulator Jacobian matrix.

To derive a simple expression for the manipulator Jacobian matrix, it is helpful to first partition it into two $3 \times n$ blocks, $A(q)$ and $B(q)$, as follows:

$$J(q) \triangleq \begin{bmatrix} A(q) \\ \text{------} \\ B(q) \end{bmatrix} \begin{matrix} \}3 \\ \\ \}3 \end{matrix} \tag{5-7-3}$$

The basic differential relation between tool displacement and joint displacement in Eq. (5-7-2) can then be decomposed into two separate equations, as follows:

$$dp = A(q)\, dq \tag{5-7-4}$$

$$d\phi = B(q)\, dq \tag{5-7-5}$$

Thus $A(q)$ is the part of the manipulator Jacobian associated with linear tool displacement, while $B(q)$ is the part of the manipulator Jacobian associated with angular tool displacement. To find an expression for $A(q)$, first note that if both sides of Eq. (5-7-4) are divided by an infinitesimal time displacement dt, then, recalling the notation $\dot{p} = dp/dt$ and $\dot{q} = dq/dt$, this yields:

$$\dot{p} = A(q)\dot{q} \tag{5-7-6}$$

Hence the $3 \times n$ Jacobian submatrix $A(q)$ can be interpreted as a linear transformation which maps instantaneous joint velocity $\dot{q}$ into instantaneous linear tool-tip velocity $\dot{p}$. Now differentiating the tool-tip position function $p(q)$ with respect to time yields $\dot{p} = [dp(q)/dq]\dot{q}$. Thus $A(q) = dp(q)/dq$, or:

$$A_{kj}(q) = \frac{\partial p_k(q)}{\partial q_j} \qquad 1 \leq k \leq 3,\ 1 \leq j \leq n \tag{5-7-7}$$

It is also possible to derive an alternative expression for $A(q)$ with vector cross products that does not require taking derivatives. We will use the derivative formulation in Eq. (5-7-7) because it is convenient and, moreover, it builds directly on our kinematics formulation. For an implementation of $J(q)$ on a computer, one might use the more fundamental vector cross-product representation discussed in Asada and Slotine (1986).

Next we examine the Jacobian submatrix $B(q)$ in Eq. (5-7-5). If $b^k(q)$ denotes the kth column of $B(q)$, then Eq. (5-7-5) can be rewritten in terms of the columns of $B(q)$ as follows:

$$d\phi = \sum_{k=1}^{n} (dq_k)b^k(q) \tag{5-7-8}$$

It is evident from Eq. (5-7-8) that column $b^k(q)$ represents a set of gains or constants of proportionality which relate the displacement of joint k to the angular tool displacement $d\phi$ when all other joints remain motionless. To derive an expression for $b^k(q)$, we consider two cases. First, suppose joint k is prismatic. Then from Fig. 5-5 we see that link k and all subsequent links including the tool translate along axis z^{k-1}. Since there is no angular tool displacement induced by this linear motion along axis z^{k-1}, $d\phi = 0$, which means that $b^k(q) = 0$.

Next, consider the case when joint k is revolute. Here link k and all subsequent links including the tool rotate about axis z^{k-1}. In this case the angular tool displacement induced by the motion about axis z^{k-1} is:

$$d\phi = (dq_k)z^{k-1} \tag{5-7-9}$$

Comparing Eq. (5-7-9) with Eq. (5-7-8), we conclude that $b^k(q) = z^{k-1}(q)$ when joint k is revolute. Rather than have two separate expressions for $b^k(q)$, we can combine them into a single expression by using the joint type parameter ξ_k introduced in Eq. (2-6-2). Recall that $\xi_k = 0$ if joint k is prismatic and $\xi_k = 1$ if joint k is revolute. Thus in terms of the joint type parameter ξ_k, the kth column of the manipulator Jacobian submatrix $B(q)$ is:

$$b^k(q) = \xi_k z^{k-1}(q) \qquad 1 \leq k \leq n \tag{5-7-10}$$

To complete the analysis of the manipulator Jacobian, we must find an expression for $z^{k-1}(q)$ in terms of the kinematic parameters. This can be done by using the homogeneous link-coordinate transformation matrices. First note that the coordinates of z^{k-1}, with respect to frame L_{k-1}, are simply i^3. Therefore, if we transform to the base, the coordinates of z^{k-1} with respect to the base frame are:

$$z^{k-1}(q) = R_0^{k-1}(q)i^3 \qquad 1 \leq k \leq n \tag{5-7-11}$$

We can now substitute the results of Eqs. (5-7-7) and (5-7-10) in Eq. (5-7-3). This generates the following final expression for the $6 \times n$ manipulator Jacobian matrix:

$$J(q) = \begin{bmatrix} \dfrac{\partial p(q)}{\partial q_1} & \dfrac{\partial p(q)}{\partial q_2} & \cdots & \dfrac{\partial p(q)}{\partial q_n} \\ \hline \xi_1 z^0(q) & \xi_2 z^1(q) & \cdots & \xi_n z^{n-1}(q) \end{bmatrix} = \begin{bmatrix} A(q) \\ \hline B(q) \end{bmatrix} \tag{5-7-12}$$

The computation of $z^{k-1}(q)$ in Eq. (5-7-11) can be structured in such a manner that the columns of $B(q)$ are computed recursively. The final pass produces the arm matrix $T_0^n(q)$, from which $p(q)$ and $A(q)$ can be determined using Eq. (5-7-7). We summarize this procedure in the form of the following algorithm where the 3×4 homogeneous coordinate conversion matrix H_1 was defined in Eq. (2-4-1):

Algorithm 5-7-1: Manipulator Jacobian

1. Set $T_0^0 = I$, $k = 1$.
2. Compute:

   $b^k(q) = \xi_k R_0^{k-1}(q)i^3$
3. Use Prop. 2-6-1 to compute $T_0^k(q) = T_0^{k-1}(q)T_{k-1}^k(q)$.
4. Set $k = k + 1$. If $k \leq n$, go to step 1; else, continue.
5. Compute $p(q) = H_1 T_0^n(q)i^4$ and:

   $$a^k = \frac{\partial p(q)}{\partial q_k} \qquad 1 \leq k \leq n$$
6. Form $J(q)$ using Eq. (5-7-12).

5-7-1 Manipulator Jacobian of a Four-Axis SCARA Robot (Adept One)

As an example of an application of Algorithm 5-7-1, consider the four-axis SCARA robot whose kinematic description was developed in Chap. 2. First, we apply steps 2 to 4 of Algorithm 5-7-1 starting with $T_0^0 = I$ and $k = 1$. Since joint 1 is revolute, $b^1(q) = R_0^0 i^3 = i^3$. Next, from Eq. (2-8-3) we update $T_0^k(q)$ as follows:

$$T_0^1 = T_0^0 T_0^1$$

$$= \begin{bmatrix} C_1 & S_1 & 0 & a_1C_1 \\ S_1 & -C_1 & 0 & a_1S_1 \\ 0 & 0 & -1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-7-13}$$

Since joint 2 is also revolute, we have $b^2(q) = R_0^1(q)i^3 = -i^3$. Again, we update the composite link-coordinate transformation matrix using Eq. (2-8-3):

$$T_0^2 = T_0^1 T_1^2$$

$$= \begin{bmatrix} C_{1-2} & S_{1-2} & 0 & a_1C_1 + a_2C_{1-2} \\ S_{1-2} & -C_{1-2} & 0 & a_1S_1 + a_2S_{1-2} \\ 0 & 0 & -1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-7-14}$$

The third joint is prismatic, which means that $b^3(q) = 0$. Updating the composite link-coordinate transformation matrix using Eq. (2-8-3) then yields:

$$T_0^3 = T_0^2 T_2^3$$

$$= \begin{bmatrix} C_{1-2} & S_{1-2} & 0 & a_1C_1 + a_2C_{1-2} \\ S_{1-2} & -C_{1-2} & 0 & a_1S_1 + a_2S_{1-2} \\ 0 & 0 & -1 & d_1 - q_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-7-15}$$

Since joint 4 is revolute, we have $b^4(q) = R_0^3 i^3 = -i^3$. Finally, we update $T_0^k(q)$ one last time to produce the arm matrix. Using Prop. 2-8-1, this yields:

$$T_0^4 = T_0^3 T_3^4$$

$$= \begin{bmatrix} C_{1-2-4} & S_{1-2-4} & 0 & a_1C_1 + a_2C_{1-2} \\ S_{1-2-4} & -C_{1-2-4} & 0 & a_1S_1 + a_2S_{1-2} \\ 0 & 0 & -1 & d_1 - q_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-7-16}$$

Applying step 5 of Algorithm 5-7-1 using Eq. (5-7-16) and the definition of H_1 in Eq. (2-4-1), we get the following tool-tip position function:

$$p(q) = H_1 T_0^4 i^4$$

$$= [a_1C_1 + a_2C_{1-2},\ a_1S_1 + a_2S_{1-2},\ d_1 - q_3 - d_4]^T \tag{5-7-17}$$

Taking partial derivatives of the expression in Eq. (5-7-17), the Jacobian submatrix $A(q)$ associated with linear tool displacement is:

$$A(q) = \begin{bmatrix} -a_1S_1 - a_2S_{1-2} & a_2S_{1-2} & 0 & 0 \\ a_1C_1 + a_2C_{1-2} & -a_2C_{1-2} & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{5-7-18}$$

The computation of the manipulator Jacobian for the four-axis SCARA robot is now complete, and the final result is:

$$J(q) = \left[\begin{array}{cccc} -a_1S_1 - a_2S_{1-2} & a_2S_{1-2} & 0 & 0 \\ a_1C_1 + a_2C_{1-2} & -a_2C_{1-2} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 \end{array}\right] \tag{5-7-19}$$

Recall that the first three rows of $J(q)$ correspond to linear tool displacement, while the last three rows correspond to angular tool displacement. Comparison of Eq. (5-7-19) with Eq. (5-1-8) reveals that the first three rows of $J(q)$ are identical to the first three rows of the tool-configuration Jacobian $V(q)$. This is true *in general* because the tool configuration vector is $x = (p, [\exp(q_n/\pi)]r^3)$. Hence, from Eq. (5-7-1):

$$du_k = dx_k \qquad 1 \leq k \leq 3 \tag{5-7-20}$$

However, there is a difference in the last three rows between $J(q)$ and $V(q)$. This is because $V(q)$ is based on one representation of tool orientation (a scaled version of the approach vector) while $J(q)$ is based on another representation (infinitesimal rotations). Note from Eq. (5-7-19) that the last three rows of $J(q)$ are particularly simple. This is a consequence of the fact that all four axes of the SCARA robot are *always* vertical. The third joint does not contribute to the angular displacement of the tool because it is a prismatic joint.

5-7-2 Manipulator Jacobian of a Five-Axis Articulated Robot (Rhino XR-3)

As a second example of an application of Algorithm 5-7-1, consider the five-axis articulated robot whose kinematic description was developed in Chap. 2. Although this robot is more complex than the four-axis SCARA robot, we can simplify the task of computing $J(q)$ by exploiting the fact that the first three rows of $J(q)$ and those of $V(q)$ are always identical. The tool-configuration Jacobian $V(q)$ has already been computed in Eq. (5-1-5). Thus $A(q)$ can be obtained from Eq. (5-1-5) and we need only compute $B(q)$. Now $\xi_k = 1$ for $1 \leq k \leq 5$, since all of the joints are revolute. The first column of $B(q)$ is $b^k(q) = R_0^0 i^3 = i^3$. Thus, from Eq. (5-1-5a), the first column of the manipulator Jacobian matrix of the five-axis articulated robot is:

$$j^1(q) = \left[\begin{array}{c} -S_1(a_2C_2 + a_3C_{23} + a_4C_{234} - d_5S_{234}) \\ C_1(a_2C_2 + a_3C_{23} + a_4C_{234} - d_5S_{234}) \\ 0 \\ \hline 0 \\ 0 \\ 1 \end{array}\right] \tag{5-7-21}$$

Next we compute $T_0^1 = T_0^0 T_0^1$. From the kinematic analysis in Chap. 2, and specifically Eq. (2-7-3), we have:

$$T_0^1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-7-22}$$

Thus $b^2(q) = R_0^1 i^3 = [-S_1, C_1, 0]^T$. Hence, from Eq. (5-1-5b), the second column of the manipulator Jacobian is:

$$j^2(q) = \begin{bmatrix} -C_1(a_2S_2 + a_3S_{23} + a_4S_{234} + d_5C_{234}) \\ -S_1(a_2S_2 + a_3S_{23} + a_4S_{234} + d_5C_{234}) \\ -a_2C_2 - a_3C_{23} - a_4C_{234} + d_5S_{234} \\ \hline -S_1 \\ C_1 \\ 0 \end{bmatrix} \tag{5-7-23}$$

Next we compute $T_0^2 = T_0^1 T_1^2$. From Eq. (2-7-3), this yields:

$$T_0^2 = \begin{bmatrix} C_1C_2 & -C_1S_2 & -S_1 & a_2C_1C_2 \\ S_1C_2 & -S_1S_2 & C_1 & a_2S_1C_2 \\ -S_2 & -C_2 & 0 & d_1 - a_2S_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-7-24}$$

Thus $b^3(q) = R_0^2 i^3 = [-S_1, C_1, 0]^T$. Hence, from Eq. (5-1-5c), the third column of the manipulator Jacobian is:

$$j^3(q) = \begin{bmatrix} -C_1(a_3S_{23} + a_4S_{234} + d_5C_{234}) \\ -S_1(a_3S_{23} + a_4S_{234} + d_5C_{234}) \\ -a_3C_{23} - a_4C_{234} + d_5S_{234} \\ \hline -S_1 \\ C_1 \\ 0 \end{bmatrix} \tag{5-7-25}$$

Next we compute $T_0^3 = T_0^2 T_2^3$. From Eq. (2-7-3), this yields:

$$T_0^3 = \begin{bmatrix} C_1C_{23} & -C_1S_{23} & -S_1 & C_1(a_2C_2 + a_3C_{23}) \\ S_1C_{23} & -S_1S_{23} & C_1 & S_1(a_2C_2 + a_3C_{23}) \\ -S_{23} & -C_{23} & 0 & d_1 - a_2S_2 - a_3S_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-7-26}$$

Thus $b^4(q) = R_0^3 i^3$, and we again have $b^4(q) = [-S_1, C_1, 0]^T$. Hence, from Eq. (5-1-5d), the fourth column of the manipulator Jacobian is:

$$j^4(q) = \begin{bmatrix} -C_1(a_4S_{234} + d_5C_{234}) \\ -S_1(a_4S_{234} + d_5C_{234}) \\ -a_4C_{234} + d_5S_{234} \\ \hline -S_1 \\ C_1 \\ 0 \end{bmatrix} \tag{5-7-27}$$

Finally, we compute $T_0^4 = T_0^3 T_3^4$. From Eqs. (2-7-3) and (2-7-4), after multiplication and simplification, we have:

$$T_0^4 = \begin{bmatrix} C_1C_{234} & S_1 & -C_1S_{234} & C_1(a_2C_2 + a_3C_{23} + a_4C_{234}) \\ S_1C_{234} & -C_1 & -S_1S_{234} & S_1(a_2C_2 + a_3C_{23} + a_4C_{234}) \\ -S_{234} & 0 & -C_{234} & d_1 - a_2S_2 - a_3S_{23} - a_4S_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5-7-28}$$

Thus $b^5(q) = R_0^4 i^3 = [-C_1S_{234}, -S_1S_{234}, -C_{234}]^T$. Hence, from Eq. (5-1-5e), the fifth column of the manipulator Jacobian is:

$$j^5(q) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \hline -C_1S_{234} \\ -S_1S_{234} \\ -C_{234} \end{bmatrix} \tag{5-7-29}$$

This completes the computation of the 6×5 manipulator Jacobian matrix of the five-axis articulated robot with tool pitch and tool roll motion. Note that the contribution to the angular tool displacement is identical for the shoulder, elbow, and tool pitch axes. This is a consequence of the fact that these three axes always remain *parallel* to one another in a vertically jointed robot.

5-7-3 Manipulator Jacobian of a Three-Axis Planar Articulated Robot

As a third example of an application of Algorithm 5-7-1, consider the three-axis articulated robot whose kinematic description was developed in Chap. 3. The tool-configuration Jacobian $V(q)$ of this robot has already been computed in Eq. (5-1-10). Thus we need only determine the angular tool Jacobian submatrix $B(q)$. Again the joints are all revolute, which means that $b^k(q) = z^{k-1}(q)$ for $1 \leq k \leq 3$. For the planar articulated robot, whose link-coordinate diagram is shown in Fig. 3-11, all three of the z axes are aligned and remain so as the robot moves. Thus, from Eq. (5-1-10), the manipulator Jacobian of the three-axis planar articulated robot is:

$$J(q) = \left[\begin{array}{ccc} -a_1S_1 - a_2S_{12} & -a_2S_{12} & 0 \\ a_1C_1 + a_2C_{12} & a_2C_{12} & 0 \\ 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{array}\right] \tag{5-7-30}$$

5-8 INDUCED JOINT TORQUES AND FORCES

The manipulator Jacobian $J(q)$, like the tool-configuration Jacobian $V(q)$, can be used for resolved-motion rate control (Whitney, 1972). However, $J(q)$ can also be used to relate external forces and moments applied at the end of the arm to internal torques and forces induced at the joints. To derive an expression for the induced joint torques and forces, let $F^{\text{tool}} \in \mathbf{R}^6$ denote the vector of external forces and moments applied at the tool tip due to either a load or contact with the environment:

$$F^{\text{tool}} \triangleq \left[\begin{array}{c} f^{\text{tool}} \\ \hline n^{\text{tool}} \end{array}\right] \tag{5-8-1}$$

Here f^{tool} is a 3×1 end-of-arm *force vector* acting at the tool tip, while n^{tool} is a 3×1 end-of-arm *moment vector* as shown in Fig. 5-4. Next, let $\tau \in \mathbf{R}^n$ denote the vector of *joint torques and forces*. If joint k is revolute, then τ_k represents the torque about axis k produced by the actuator; whereas if joint k is prismatic, τ_k represents the force along axis k produced by the actuator. The relationship between f^{tool} and τ is a simple and elegant one, as can be seen from the following fundamental result:

Proposition 5-8-1: Equivalent Joint Torques and Forces. Let $J(q)$ be the manipulator Jacobian matrix of a robotic arm with frictionless joints. Then the internal joint torque and force vector τ induced by an external end-of-arm force and moment vector F^{tool} is:

$$\tau = J^T(q)F^{\text{tool}}$$

Proof. We use the principle of *virtual work*, which is based on instantaneous infinitesimal displacements of the tool (Craig, 1986). The work performed, which has units of energy, must be the same regardless of the coordinate system within which it is measured or expressed. The virtual work done by an infinitesimal displacement du of the tool is $dW = F^{\text{tool}} \cdot du$. Alternatively, the virtual work done by the corresponding infinitesimal displacement dq of the joints is $dW = \tau \cdot dq$. By the principle of virtual work, these two formulations of the work performed are equal:

$$(F^{\text{tool}})^T du = \tau^T dq$$

Here we have represented the dot products in the virtual work equation using

the transpose operation. Next, recall from Eq. (5-7-2) that the tool displacement du is related to the joint displacement dq through the manipulator Jacobian matrix as follows:

$$du = J(q)dq$$

If we now substitute this expression for du in the virtual work equation, the result is:

$$(F^{\text{tool}})^T J(q)\, dq = \tau^T\, dq.$$

But this must hold for all infinitesimal joint displacements dq. Therefore, $(F^{\text{tool}})^T J(q) = \tau^T$, and taking transposes of both sides yields:

$$\tau = J^T(q)F^{\text{tool}}$$

Thus the transpose of the manipulator Jacobian maps the external force and moment vector F^{tool} into the equivalent internal joint torque and force vector τ needed to keep the manipulator in static equilibrium. Refer to Fig. 5-7 for a pictorial representation of Prop. 5-8-1. Note that the equation in Prop. 5-8-1 does not include loading torques and forces on the individual joints induced by gravity. The effects of gravity are analyzed separately in Chap. 6.

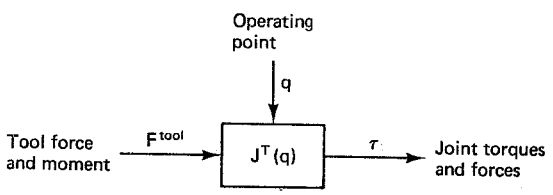


Figure 5-7 Induced joint torques and forces.

The relationship between F^{tool} and τ in Prop. 5-8-1 can be turned around to our advantage. Indeed, if the induced torque and force vector τ can be *measured* with sensors placed at the joints, then the associated end-of-arm force and moment vector F^{tool} can be *inferred* by solving the equation in Prop. 5-8-1 for F^{tool}. In this way, once the tool is brought into contact with an object in the environment, the actuator torques and forces needed to establish a desired force and moment vector at the tool tip can be computed using Prop. 5-8-1. This type of control, which is called *active compliance*, is potentially quite useful for assembly operations such as part insertion and part mating tasks. However, certain practical difficulties arise in actual implementations due to such factors as static friction and elastic deformation of the links.

Example 5-8-1: Equivalent Joint Torques and Forces

As an example of an application of Prop. 5-8-1, consider the four-axis SCARA robot whose manipulator Jacobian was computed in Eq. (5-7-19). Applying Prop. 5-8-1, we have:

$$\tau = \begin{bmatrix} -a_1S_1 - a_2S_{1-2} & a_1C_1 + a_2C_{1-2} & 0 & 0 & 0 & 1 \\ a_2S_{1-2} & -a_2C_{1-2} & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} F^{\text{tool}} \qquad (5\text{-}8\text{-}2)$$

Recall from Eq. (5-8-1) that the end-of-arm force and moment vector F^{tool} consists of three linear force components f^{tool} augmented by three angular moment components n^{tool}. Multiplying Eq. (5-8-2) out, we have:

$$\tau_1 = (-a_1 S_1 - a_2 S_{1-2}) f_1^{\text{tool}} + (a_1 C_1 + a_2 C_{1-2}) f_2^{\text{tool}} + n_3^{\text{tool}} \tag{5-8-3a}$$

$$\tau_2 = a_2 S_{1-2} f_1^{\text{tool}} - a_2 S_{1-2} f_2^{\text{tool}} - n_3^{\text{tool}} \tag{5-8-3b}$$

$$\tau_3 = -f_3^{\text{tool}} \tag{5-8-3c}$$

$$\tau_4 = -n_3^{\text{tool}} \tag{5-8-3d}$$

Note from Eq. (5-8-3c) that τ_3 represents a joint force because joint 3 is prismatic, while the remaining components of τ represent joint torques. Observe that the vertical component of the end-of-arm force f^{tool} must be compensated for exclusively by joint 3, while the horizontal components of f^{tool} are compensated for by joints 1 and 2 collectively. The vertical component of the external end-of-arm moment n^{tool} is compensated for by joints 1, 2, and 4. Since the horizontal components of n^{tool} do not appear in the right-hand side of Eq. (5-8-3), this means that these components of n^{tool} must be borne by the physical *structure* of the manipulator, rather than by the robot actuators.

5-8-1 Tool Compliance and Stiffness

When an external force and moment vector F^{tool} is applied to the end of the arm, a displacement of the tool will occur. The amount of displacement or deflection will depend on both the magnitude of F^{tool} and the "stiffness" of the arm. A well-designed manipulator must have considerable stiffness at the end of the arm, because otherwise the load carried by the manipulator will cause an unacceptable deflection of the tool tip.

There are several sources of tool displacement (Asada and Slotine, 1986). One is due to elastic deflections of the links. This can be significant when the links are very long, as is the case, for example, with the space shuttle arm. However, most industrial robots are designed to have links which are quite rigid. Perhaps the most significant source of tool displacement lies in the joint drive system. Each joint controller will produce a restoring torque or force whenever there is an error, or displacement, between the desired position of the link and the measured position. For *small joint displacements* Δq, the relationship between the displacement and the restoring torque or force $\Delta\tau$ that the controller generates can be modeled as follows for each joint:

$$\Delta\tau_k = \gamma_k \Delta q_k \qquad 1 \le k \le n \tag{5-8-4}$$

Here $\gamma_k > 0$ can be thought of as a spring constant for joint k. It will depend on the gain of the feedback control system used to regulate the position of link k. The n individual spring constants together form the following $n \times n$ diagonal matrix Γ, called the *joint stiffness matrix* of the manipulator:

$$\Gamma \triangleq \text{diag}\,\{\gamma_1, \gamma_2, \cdots, \gamma_n\} \tag{5-8-5}$$

Note that as long as each joint has some nonzero restoring torque or force, the joint stiffness matrix Γ will be invertible. Although joint stiffness is relatively easy to

characterize, it is the tool stiffness at the end of the arm that is of primary interest. This can be expressed in terms of the joint stiffness and the manipulator Jacobian, as can be seen from the following result:

Proposition 5-8-2: Tool Compliance. Let $J(q)$ and Γ be the manipulator Jacobian matrix and the joint stiffness matrix, respectively, of a robotic arm with rigid links. If an external force and moment vector F^{tool} is applied to the end of the arm, and if the resulting tool deflection Δu is infinitesimal, then:

$$\Delta u = J(q)\Gamma^{-1}J^T(q)F^{\text{tool}}$$

Proof. From Prop. 5-8-1, we have $\Delta\tau = J^T(q)F^{\text{tool}}$. But Eqs. (5-8-4) and (5-8-5) imply $\Delta\tau = \Gamma\,\Delta q$. Thus, $\Gamma\,\Delta q = J^T(q)F^{\text{tool}}$. Since Γ is nonsingular, one can solve for Δq as follows:

$$\Delta q = \Gamma^{-1}J^T(q)F^{\text{tool}}$$

From the definition of $J(q)$ in Eq. (5-7-2), we have $du = J(q)\,dq$, where du and dq represent infinitesimal displacements of the tool and joints, respectively. If the joint deflection Δq and tool deflection Δu are sufficiently small (approaching infinitesimal), then we can equate Δu with du and Δq with dq, and this yields:

$$\begin{aligned}\Delta u &= J(q)\,\Delta q \\ &= J(q)\Gamma^{-1}J^T(q)F^{\text{tool}}\end{aligned}$$

Note that Prop. 5-8-2, which is illustrated in Fig. 5-8, is applicable only to external end-of-arm force and moment vectors which cause a *small* displacement of the tool.

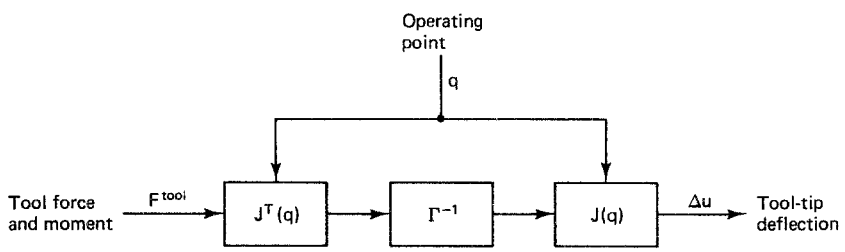


Figure 5-8 Deflection of tool tip.

The utility of Prop. 5-8-2 lies in the fact that it provides a measure of the *sensitivity* of the arm to deflection at various points q in the joint space. Let the 6×6 coefficient matrix in Prop. 5-8-2 be denoted $G(q)$. That is,

$$G(q) \triangleq J(q)\Gamma^{-1}J^T(q) \tag{5-8-6}$$

We call $G(q)$ the tool *compliance matrix* of the manipulator. Note that for a general manipulator ($n = 6$) or for a kinematically redundant manipulator ($n > 6$), the manipulator Jacobian $J(q)$ will have full row rank as long as q is not a joint-space singu-

larity. Under these conditions, the tool compliance matrix $G(q)$ is invertible, and from Eq. (5-8-6) and Prop. 5-8-2:

$$F^{\text{tool}} = G^{-1}(q)\,\Delta u \tag{5-8-7}$$

The inverse of the tool compliance matrix is called the tool *stiffness matrix*. Here Eq. (5-8-7) is analogous to the joint stiffness equation, Eq. (5-8-4). When the manipulator is near a joint-space singularity, the determinant of the tool compliance matrix is very small and the components of the tool stiffness matrix become very large. Thus a robot becomes "stiff" near a singularity.

Example 5-8-2: Tool Deflection

As an example of an application of Prop. 5-8-2, consider the four-axis SCARA robot whose manipulator Jacobian was computed in Eq. (5-7-19). To simplify the equations somewhat, suppose each joint has the same spring constant $\gamma > 0$, in which case the joint stiffness matrix is simply $\Gamma = \gamma I$. Next suppose an end-of-arm force and moment vector F^{tool} is applied at the tool tip. From Prop. 5-8-2, the resulting tool deflection is:

$$\Delta u = J(q)\Gamma^{-1}J^T(q)F^{\text{tool}}$$

$$= \frac{J(q)J^T(q)F^{\text{tool}}}{\gamma}$$

$$= \begin{bmatrix} -a_1S_1 - a_2S_{1-2} & a_2S_{1-2} & 0 & 0 \\ a_1C_1 + a_2C_{1-2} & -a_2C_{1-2} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} -a_1S_1 - a_2S_{1-2} & a_1C_1 + a_2C_{1-2} & 0 & 0 & 0 & 1 \\ a_2S_{1-2} & -a_2C_{1-2} & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \frac{F^{tool}}{\gamma}$$

$$= G(q)F^{\text{tool}}$$

The matrix multiplication defining $G(q)$ can now be performed, and the resulting equations for the individual components of Δu can be written out separately in terms of the force and moment parts of F^{tool}. The final result is:

$$\Delta p_1 = \frac{[(a_1S_1 + a_2S_{1-2})^2 + (a_2S_{1-2})^2]f_1^{\text{tool}}}{\gamma} - \frac{(a_1S_1 + 2a_2S_{1-2})n_3^{\text{tool}}}{\gamma}$$
$$- \frac{[(a_1S_1 + a_2S_{1-2})(a_1C_1 + a_2C_{1-2}) + a_2^2S_{1-2}C_{1-2}]f_2^{\text{tool}}}{\gamma}$$

$$\Delta p_2 = \frac{[(a_1C_1 + a_2C_{1-2})^2 + (a_2C_{1-2})^2]f_2^{\text{tool}}}{\gamma} + \frac{(a_1C_1 + 2a_2C_{1-2})n_3^{\text{tool}}}{\gamma}$$
$$- \frac{[(a_1S_1 + a_2S_{1-2})(a_1C_1 + a_2C_{1-2}) + a_2^2S_{1-2}C_{1-2}]f_1^{\text{tool}}}{\gamma}$$

$$\Delta p_3 = \frac{f_3^{\text{tool}}}{\gamma}$$

$$\Delta\phi_1 = 0$$

$$\Delta\phi_2 = 0$$

$$\Delta\phi_3 = -\frac{(a_1S_1 + 2a_2S_{1-2})f_1^{\text{tool}}}{\gamma} + \frac{(a_1C_1 + 2a_2C_{1-2})f_2^{\text{tool}}}{\gamma} + \frac{3n_3^{\text{tool}}}{\gamma}$$

Notice that there are no angular tool deflections $\Delta\phi_1$ and $\Delta\phi_2$, because the horizontal moments n_1^{tool} and n_2^{tool} are supported by the (presumably) rigid physical structure of the SCARA robot. The vertical component of the force vector causes a tool displacement Δp_3 that is proportional to $-f_3^{\text{tool}}$ with the constant of proportionality being $1/\gamma$, the compliance of the prismatic joint. The horizontal deflections of the tool and the angular deflection of the tool about the z axis are the most complex because three of the four joints contribute to these tool displacements.

5-8-2 Joint-Space Singularities

Recall that a joint-space singularity is a point q at which the Jacobian matrix loses full rank. To provide a simple physical interpretation of joint-space singularities, we use the concepts of the range space and the null space of a matrix discussed in Chap. 1. Recall that for a matrix A, rank (A) is the dimension of the range space $R(A)$ and nullity (A) is the dimension of the null space $N(A)$. From Eq. 5-7-2, if du denotes an infinitesimal displacement of the tool and dq denotes the corresponding infinitesimal displacement of the joints, then:

$$du = J(q)\, dq \tag{5-8-8}$$

Now $J(q)$ is a $6 \times n$ matrix. If $n < 6$, or if the manipulator is at a joint-space singularity, then rank $[J(q)] < 6$. This implies that at the point q there exist infinitesimal tool displacements du that are not physically realizable by *any* joint displacement dq. Hence there are forbidden or unrealizable *directions* in which the tool cannot move starting from point q. If the point q is a boundary singularity, then this much is obvious, because the tool clearly cannot penetrate the surface of the work envelope. However, unrealizable displacements of the tool can also occur inside the work envelope at interior singularities. This interpretation of a joint-space singularity is summarized in Fig. 5-9, where the forbidden directions are vectors in $\mathbf{R}^6$ that lie outside of $R[J(q)]$, the range space of $J(q)$.

An alternative interpretation of what happens when $J(q)$ loses full rank arises

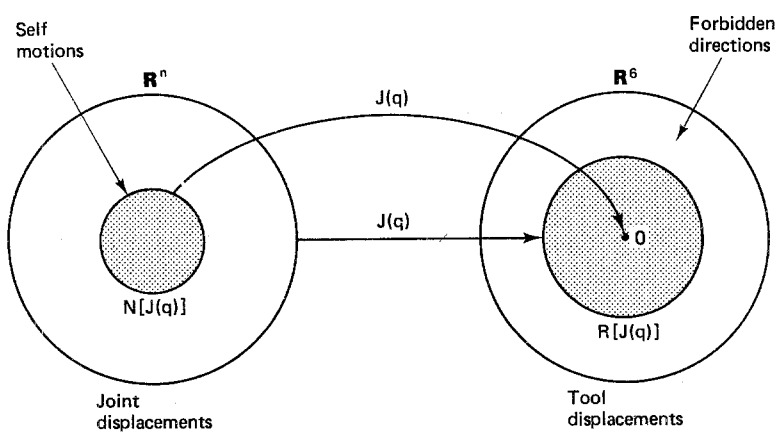


Figure 5-9 Self-motions and forbidden tool displacements.

from an analysis of the null space of $J(q)$. If rank $[J(q)] < n$, then, from Eq. (1-5-9), nullity $[J(q)] > 0$. It then follows from Eq. (5-8-8) that there exist joint displacements dq which cause a *null* tool displacement, $du = 0$. These joint displacements which produce no movement of the tool are referred to as *self-motions* of the manipulator, as illustrated in Fig. 5-9. In addition to occurring at joint-space singularities, such as those discussed in Example 5-2-2, self-motions also arise in kinematically redundant robots. This is a consequence of the fact that, for a $6 \times n$ manipulator Jacobian matrix $J(q)$:

$$\text{nullity } [J(q)] \geq n - 6 \tag{5-8-9}$$

Consequently, whenever $n > 6$, there are nonzero vectors in $N[J(q)]$, the null space of $J(q)$. These nonzero vectors generate self-motions of the manipulator as shown in Fig. 5-9 and illustrated previously in Example 5-2-2.

The interpretation of joint-space singularities in terms of joint displacements and tool displacements has a counterpart in terms of external end-of-arm force and moment vectors and induced joint torques and forces. Recall from Prop. 2-8-1 that if F^{tool} denotes an end-of-arm force and moment vector and τ denotes the corresponding joint torque and force vector, then:

$$\tau = J^T(q)F^{\text{tool}} \tag{5-8-10}$$

At a joint-space singularity, $J^T(q)$ loses full rank because rank (J^T) = rank (J). Now if rank $[J^T(q)] < 6$, then nullity $[J^T(q)] > 0$. Hence there are nonzero force and moment vectors that induce no joint torques or forces. That is, if q is a joint-space singularity, then there are *directions* in which the manipulator tool cannot exert a force or moment when it is in contact with the environment no matter what joint torques and forces are applied. When external forces and moments are applied in these directions, they must be borne entirely by the structure of the manipulator, as indicated in Fig. 5-10, rather than by the joint actuators.

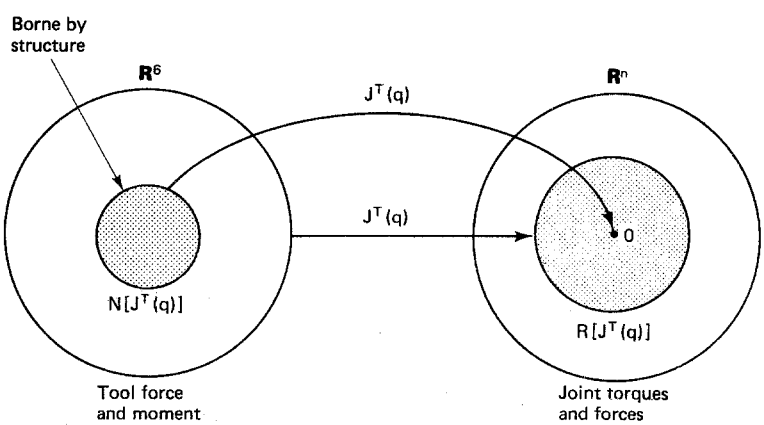


Figure 5-10 Tool forces and moments supported by manipulator structure.

Example 5-8-3: Joint-Space Singularities

Consider again the four-axis SCARA robot whose manipulator Jacobian was computed in Eq. (5-7-19). Applying Prop. 5-8-1, we have:

$$\tau = \begin{bmatrix} -a_1S_1 - a_2S_{1-2} & a_1C_1 + a_2C_{1-2} & 0 & 0 & 0 & 1 \\ a_2S_{1-2} & -a_2C_{1-2} & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} F^{\text{tool}} \tag{5-8-11}$$

To locate the joint-space singularities, we must find those values of q at which $J(q)$ or $J^T(q)$ loses full rank. The only components of $J^T(q)$ in Eq. (5-8-11) that vary are those in the 2×2 submatrix in the upper left corner. The determinant of this submatrix is:

$$\begin{aligned} \Delta &= (-a_1S_1 - a_2S_{1-2})(-a_2C_{1-2}) - a_2S_{1-2}(a_1C_1 + a_2C_{1-2}) \\ &= a_1a_2S_1C_{1-2} + a_2^2S_{1-2}C_{1-2} - a_1a_2C_1S_{1-2} - a_2^2S_{1-2}C_{1-2} \\ &= a_1a_2(S_1C_{1-2} - C_1S_{1-2}) \\ &= a_1a_2S_2 \end{aligned} \tag{5-8-12}$$

Thus joint-space singularities occur whenever the elbow joint angle q_2 is an integer multiple of π. Suppose $q_2 = 0$, which corresponds to the arm reaching straight out as shown in the overhead view displayed in Fig. 5-11. When $q_2 = 0$, we have $S_{1-2} = S_1$ and $C_{1-2} = C_1$. Thus Eq. (5-8-11) simplifies in this case to:

$$\tau = \begin{bmatrix} -(a_1 + a_2)S_1 & (a_1 + a_2)C_1 & 0 & 0 & 0 & 1 \\ a_2S_1 & -a_2C_1 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} F^{\text{tool}} \tag{5-8-13}$$

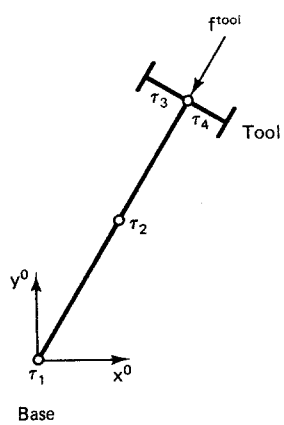


Figure 5-11 A joint-space singularity of a SCARA robot.

Observe that the first two columns of $J^T(q)$ are now proportional to one another. In particular, if C_1 times the first column is added to S_1 times the second column, the result is a column of zeros. Recall from Example 5-8-1 that because rank $[J^T(q)] < 6$, there were already force and moment vectors which did not induce joint torques or forces. In particular, because of the presence of two zero columns, an end-of-arm

force and moment vector of the following form has to be borne entirely by the physical structure of the manipulator:

$$F^{\text{tool}} = [0, 0, 0, n_1^{\text{tool}}, n_2^{\text{tool}}, 0]^T \tag{5-8-14}$$

At the joint-space singularity $q_2 = 0$, the rank of $J^T(q)$ reduces still further from 4 to 3. Therefore at this locus of points (the other components of q are not constrained) there must be an additional end-of-arm force and moment vector which induces no joint torques or forces. The key to finding F^{tool} is to make the ratio of the first two components of F^{tool} equal to the ratio of the first two columns of $J^T(q)$ in Eq. (5-8-13). For example:

$$F^{\text{tool}} = [-C_1, -S_1, 0, n_1^{\text{tool}}, n_2^{\text{tool}}, 0]^T \tag{5-8-15}$$

Substitution of F^{tool} from Eq. (5-8-15) in Eq. (5-8-13) verifies that F^{tool} induces no joint torques or forces. Thus F^{tool} is a vector in the null space of $J^T(q)$. The force part of F^{tool} in Eq. (5-8-15) is shown in Fig. 5-11 acting on the tool. It is evident from inspection that f^{tool} will not induce a torque in joint 1, 2, or 4. Furthermore, since f^{tool} is purely horizontal, it will also not induce a force in joint 3. Thus a radial reaction force from the environment must be borne entirely by the structure of the manipulator when the manipulator is in this extended singular configuration.

5-9 PROBLEMS

5-1. Consider the four-axis cylindrical-coordinate robot with tool roll motion shown in Fig. 5-12. Here the vector of joint variables is $q = [\theta_1, d_2, d_3, \theta_4]^T$. The tool-configuration function for this robot is:

$$w(q) = \begin{bmatrix} C_1 q_3 \\ S_1 q_3 \\ q_2 - d_4 \\ \hline 0 \\ 0 \\ -\exp(q_4/\pi) \end{bmatrix}$$

Find the tool-configuration Jacobian matrix $V(q)$ of this robot.

5-2. Find the joint-space singularities of the cylindrical-coordinate robot. Are there any self-motions of the manipulator at these singularities? If so, describe them.

5-3. Consider the following 3×2 matrix A:

$$A = \begin{bmatrix} 1 & 0 \\ 3 & -1 \\ 0 & 1 \end{bmatrix}$$

Find the pseudoinverse A^+.

5-4. Consider the following $m \times n$ matrix A where $m > n$:

$$A = \begin{bmatrix} B \\ \hline C \end{bmatrix}$$

Use Exercise 5-1-2 and Prop. 5-6-1 to show that if the $n \times n$ submatrix B is nonsingular, then $A^{(1)} = [B^{-1}, 0]$.

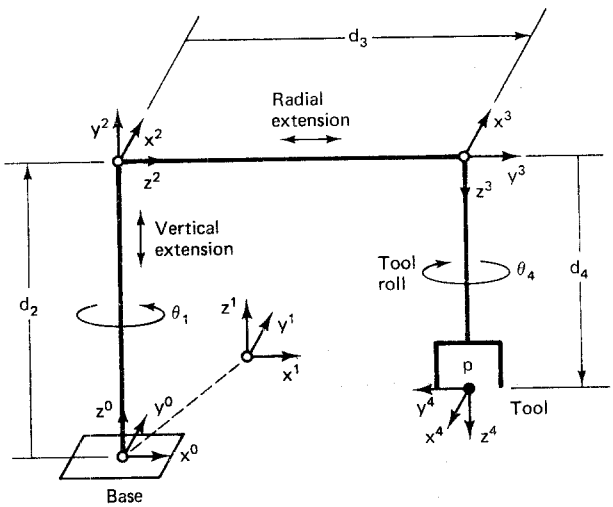


Figure 5-12 Link coordinates of a four-axis cylindrical robot.

5-5. Find the smallest least-squares solution of the following underdetermined system of equations. What is $\|x\|$?

$$x_1 - 2x_2 + 3x_3 = 2$$

$$3x_1 - 2x_2 + x_3 = -2$$

5-6. Use the method of $\{1\}$-inverses to find a general solution to the system of equations in Prob. 5-5.

5-7. Find the manipulator Jacobian matrix $J(q)$ of the four-axis cylindrical coordinate robot in Fig. 5-12.

5-8. Suppose the cylindrical-coordinate robot is not at a joint-space singularity. Find an example of an infinitesimal tool displacement vector du that is impossible to achieve because the robot has only four axes.

5-9. Suppose the cylindrical-coordinate robot is not at a joint-space singularity. Find an example of an end-of-arm force and moment vector $F^{\text{tool}} \neq 0$ that does not induce any joint torques or forces.

5-10. For the cylindrical-coordinate robot, compute the tool deflection Δu caused by an end-of-arm force and moment vector F^{tool}. Write the final result as six equations expressing Δp and $\Delta\phi$ as functions of f^{tool} and n^{tool}.

5-11. Suppose the cylindrical-coordinate robot is in the following hyperplane in joint space: $q = [q_1, q_2, 0, q_4]^T$. Describe, mathematically, the types of infinitesimal tool-tip translations dp that are possible.

5-12. Suppose the cylindrical-coordinate robot is in the following hyperplane in joint space: $q = [q_1, q_2, 0, q_4]^T$. Find a nonzero horizontal end-of-arm force vector $f^{\text{tool}} = [f_1^{\text{tool}}, f_2^{\text{tool}}, 0]^T$ that does not induce any joint torques or forces.

5-13. Consider the five-axis spherical-coordinate robot with tool pitch and tool roll motion shown in Fig. 5-13. Here the vector of joint variables is $q = [\theta_1, \theta_2, d_3, \theta_4, \theta_5]^T$. The tool-configuration function for this robot is:

$$w(q) = \begin{bmatrix} /C_1(S_2 q_3 - S_{24} d_5) \\ S_1(S_2 q_3 - S_{24} d_5) \\ d_1 + C_2 q_3 - C_{24} d_5 \\ \hline -[\exp(q_5/\pi)]C_1 S_{24} \\ -[\exp(q_5/\pi)]S_1 S_{24} \\ -[\exp(q_5/\pi)]C_{24} \end{bmatrix}$$

Find the tool-configuration Jacobian matrix $V(q)$ of this robot.

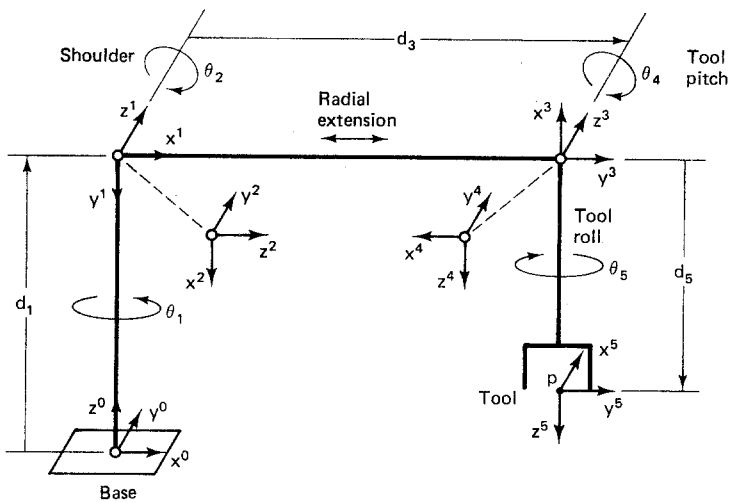


Figure 5-13 Link coordinates of a five-axis spherical robot.

5-14. Show that the spherical-coordinate robot has a joint-space singularity at $q = [q_1, 0, q_3, -\pi, q_5]^T$. Sketch the robot in this configuration. Which axes are collinear?

5-15. Find the manipulator Jacobian matrix $J(q)$ of the five-axis spherical-coordinate robot.

5-16. Find the manipulator Jacobian matrix $J(q)$ of the two-axis planar articulated robot shown in Fig. 5-14.

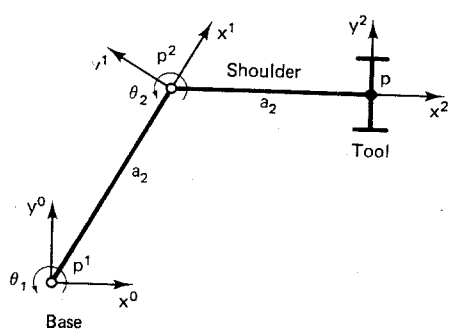


Figure 5-14 Link coordinates of a two-axis planar articulated robot.

5-17. Find the tool compliance matrix $G(q)$ of the two-axis planar articulated robot in Fig. 5-14 assuming $\Gamma = \gamma I$.

5-18. Find the tool deflection equations for the two-axis planar articulated robot in Fig. 5-14 assuming $\Gamma = \gamma I$.

5-19. Let $x = p \in \mathbf{R}^2$ represent the tool configuration of the two-axis planar articulated robot in Fig. 5-14. Find the resolved-motion rate-control equations associated with the tool-tip trajectory $x(t)$.

REFERENCES

ASADA, H., and J. E. SLOTINE (1986). *Robot Analysis and Control,* Wiley: New York.

ATHANS, M., and P. L. FALB (1966). *Optimal Control: An Introduction to the Theory and Its Applications,* McGraw-Hill: New York.

BEN-ISRAEL, A., and T. N. E. GREVILLE (1974). *Generalized Inverses: Theory and Applications,* Wiley-Interscience: New York.

CRAIG, J. (1986). *Introduction to Robotics: Mechanics and Control,* Addison-Wesley: Reading, Mass.

LOVASS-NAGY, V., R. J. MILLER, and D. L. POWERS (1978). "An introduction to the application of the simplest matrix-generalized inverse in systems science," *IEEE Trans. Circuits Syst.*, Vol. CAS-25, No. 9, p. 766.

LOVASS-NAGY, V., and R. J. SCHILLING (1987). "Control of kinematically redundant robots using {1}-inverses," *IEEE Trans. Syst. Man, Cybern.*, Vol. SMC-17, No. 4, pp. 644–649.

WHITNEY, D. E. (1969). "Resolved motion rate control of manipulators and human prostheses," *IEEE Trans. Man-Machine Syst.*, Vol. MMS-10, No. 2, pp. 47–53.

WHITNEY, D. E. (1972). "The mathematics of coordinated control of prosthetic arms and manipulators," *Trans. ASME J. Dynamic Systems, Measurement and Control,* Vol. 122, pp. 303–309.

YOSHIKAWA, T. (1984). "Analysis and control of robot manipulators with redundancy," *Proc. Robotics Research: 1st Int. Symp.*, M. Brady and R. Paul, Eds., Cambridge, Mass.: MIT Press, pp. 735–748.

6

Manipulator Dynamics

Optimal performance can be obtained from a robotic manipulator if sophisticated control strategies are employed. However, precise control of high-speed motion requires the use of a realistic dynamic model of the arm. The dynamic equations of motion of a general six-axis manipulator can be rather complex. We therefore illustrate the derivation of the equations of motion using relatively simple robotic arms. Two basic approaches are presented. The first is the Lagrange-Euler formulation, which is based on the concepts of generalized coordinates, energy, and generalized force. This approach has the advantage that each of the terms in the final closed-form equation has a simple physical interpretation in terms of such things as manipulator inertia, gravity, friction, and Coriolis and centrifugal forces. Complete dynamic models for two robots, a two-axis planar articulated arm, and a three-axis SCARA type arm, are derived.

Next an alternative approach, called the recursive Newton-Euler formulation, is presented. This approach has the advantage that it is very amenable to computer implementation, and furthermore it is computationally more efficient than the Lagrange-Euler formulation, particularly as the number of axes increases. A third example of a dynamic model, a simple one-axis arm, is derived using both the Lagrange-Euler method and the recursive Newton-Euler method. The treatment of robot arm dynamics presented here is patterned primarily after discussions found in Asada and Slotine (1986) and Fu et al. (1987). Additional investigations of robot arm dynamics can be found in Brady et al. (1982), Craig (1986), Paul (1981), and Wolovich (1987).

6-1 LAGRANGE'S EQUATION

Complex dynamic systems can be modeled in a relatively simple, elegant fashion using an approach called the *Lagrangian formulation*. The Lagrangian formulation is based on the notion of generalized coordinates, energy, and generalized forces. For an n-axis robotic arm, an appropriate set of generalized coordinates is the vector of n joint variables q. Recall that the components of q represent the joint angles of revolute joints and the joint distances of prismatic joints. Let T and U represent the *kinetic energy* and *potential energy* of the arm, respectively. We then define the *Lagrangian function* as the difference between the kinetic and potential energy as follows:

$$L(q, \dot{q}) \triangleq T(q, \dot{q}) - U(q) \tag{6-1-1}$$

Here $\dot{q} = dq/dt$ is the vector of joint velocities. Note that the kinetic energy T depends on both the position and the velocity of the arm, while the potential energy U depends on only the arm position. The general equations of motion of a robotic arm can be formulated in terms of the Lagrangian function as follows (Torby, 1984):

$$\frac{d}{dt}\frac{\partial}{\partial \dot{q}_i} L(q, \dot{q}) - \frac{\partial}{\partial q_i} L(q, \dot{q}) = F_i \qquad 1 \leq i \leq n \tag{6-1-2}$$

Here F_i is the *generalized force* acting on the ith joint. It is the residual force left over after the effects of inertial forces and gravity have been removed. The Lagrangian formulation of robot arm dynamics in Eq. (6-1-2) consists of a system of n second-order nonlinear differential equations in the vector of joint variables q. To specify these equations in more detail, we must formulate expressions for the kinetic energy T, the potential energy U, and the generalized force F.

6-2 KINETIC AND POTENTIAL ENERGY

The most complicated term in the Lagrangian function of a robotic arm is the total kinetic energy of the arm $T(q, \dot{q})$. To obtain an expression for the total kinetic energy, we begin by examining the kinetic energy of the kth link of the arm shown in Fig. 6-1.

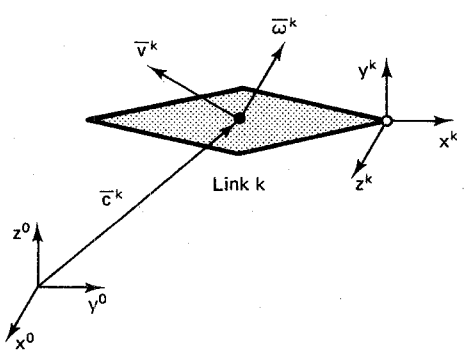


Figure 6-1 Motion of link k.

The kth link will be moving in the three-dimensional space with both a linear velocity and an angular velocity. Let $\bar{v}^k \in \mathbf{R}^3$ denote the *linear velocity* of the center of mass of link k with respect to the robot base frame, and let $\bar{\omega}^k \in \mathbf{R}^3$ denote the *angular velocity* about the center of mass with respect to the base frame. For convenience, *all* quantities are expressed with respect to the robot base frame L_0, unless specifically noted otherwise. The total kinetic energy of the arm is the sum of the kinetic energies of its links:

$$T(q, \dot{q}) = \sum_{k=1}^{n} \frac{(\bar{v}^k)^T m_k \bar{v}^k + (\bar{\omega}^k)^T D_k \bar{\omega}^k}{2} \tag{6-2-1}$$

Here m_k represents the *mass* of link k, and D_k is the 3×3 *inertia tensor* of link k about its center of mass expressed with respect to the base frame. To formulate the equations of motion of the arm we must express the kinetic energy as an explicit function of q and $\dot{q}$. We begin by examining the link inertia tensor D_k.

6-2-1 Link Inertia Tensor

An inertia tensor is a 3×3 matrix which characterizes the distribution of mass of a rigid object. To express the inertia tensor with respect to the base coordinate frame, we first consider the case of representing an inertia tensor of a rigid object with respect to a frame whose origin is located at the center of mass of the object. Let ρ denote the *mass density* of the rigid object, and let V represent the volume occupied by the object. Then the inertia tensor of the object about its center of mass, expressed with respect to a coordinate frame $L_c = \{x^c, y^c, z^c\}$ located at the center of mass, is:

$$D_c \triangleq \begin{bmatrix} \int_V (y^2 + z^2)\rho \, dV & -\int_V xy\rho \, dV & -\int_V xz\rho \, dV \\ -\int_V xy\rho \, dV & \int_V (x^2 + z^2)\rho \, dV & -\int_V yz\rho \, dV \\ -\int_V xz\rho \, dV & -\int_V yz\rho \, dV & \int_V (x^2 + y^2)\rho \, dV \end{bmatrix} \tag{6-2-2}$$

The integrals in D_c are three-dimensional integrals over the volume V occupied by the object. Note that D_c is a *symmetric matrix* containing six independent terms. The three diagonal terms are called *moments of inertia,* while the three distinct off-diagonal terms are called *products of inertia*. The moments of inertia are clearly positive quantities, but the products of inertia can be either positive or negative. If the axes of the coordinate frame L_c are aligned with the principal axes of the object, then the products of inertia will be zero. In this case, the inertia tensor is a *diagonal matrix,* and the three diagonal elements are called the *principal* moments of inertia. Expressions for the principal moments of inertia of simple geometric shapes can be found in Appendix 2.

Example 6-2-1: Inertia Tensor

As an example of an inertia tensor of a rigid object, consider the $b \times c$ rectangular rod of length a and mass m shown in Fig. 6-2. If the rod is homogeneous, then the mass density is constant and equal to $\rho = m/(abc)$. Since the axes of the L_c coordinate frame are aligned with the principal axes of the rod, the products of inertia are all zero. The three principal moments of inertia can be obtained by referring to Appendix 2, and the resulting diagonal inertia tensor is:

$$D_c = m \begin{bmatrix} \frac{b^2 + c^2}{12} & 0 & 0 \\ 0 & \frac{a^2 + c^2}{12} & 0 \\ 0 & 0 & \frac{a^2 + b^2}{12} \end{bmatrix}$$

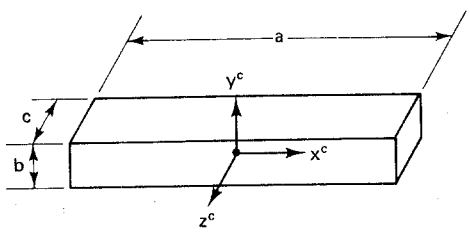


Figure 6-2 A rectangular rod.

The rectangular rod in Fig. 6-2 has a relatively simple geometry. For objects which are of an irregular shape, computing the inertia tensor using volumetric integration as in Eq. (6-2-2) may be quite difficult. In these cases, one can instead *measure* the inertia tensor experimentally (Klafter et al., 1989).

The link inertia tensor D_k in the expression for kinetic energy in Eq. (6-2-1) is the inertia tensor of link k about its center of mass expressed relative to the robot base frame L_0. That is, D_k is the inertia tensor obtained by translating the base frame to the center of mass of link k and then applying Eq. (6-2-2). Recall that the D-H algorithm assigns coordinate frame $L_k = \{x^k, y^k, z^k\}$ to the end of link k. Frame L_k and frame L_0 are related by the composite homogeneous coordinate transformation matrix $T_0^k(q)$ where:

$$T_0^k(q) = \left[\begin{array}{c|c} R_0^k(q) & p^k(q) \\ \hline 0 \quad 0 \quad 0 & 1 \end{array}\right] \qquad 1 \leq k \leq n \tag{6-2-3}$$

The rotation matrix $R_0^k(q)$ represents the orientation of frame L_k relative to frame L_0, while the translation vector $p^k(q)$ represents the position of the origin of frame L_k relative to frame L_0. The kinetic energy due to the angular velocity of link k about its center of mass is expressed in Eq. (6-2-1) in terms of frame L_0 coordinates. To reexpress this energy in terms of frame L_k coordinates, we can use $R_k^0 = (R_0^k)^{-1} = (R_0^k)^T$ to convert the angular velocity $\bar{\omega}^k$ from frame L_0 coordinates

to frame L_k coordinates. This yields the following equivalent expression for the kinetic energy of link k due to its angular velocity about its center of mass:

$$\frac{(\bar{\omega}^k)^T D_k \bar{\omega}^k}{2} = \frac{(R_k^0 \bar{\omega}^k)^T \bar{D}_k (R_k^0 \bar{\omega}^k)}{2}$$
$$= \frac{(\bar{\omega}^k)^T (R_k^0)^T \bar{D}_k R_k^0 \bar{\omega}^k}{2}$$
$$= \frac{(\bar{\omega}^k)^T R_0^k \bar{D}_k (R_0^k)^T \bar{\omega}^k}{2} \qquad (6\text{-}2\text{-}4)$$

Here $\bar{D}_k$ denotes the inertia tensor of link k about its center of mass expressed with respect to coordinate frame L_k. That is, $\bar{D}_k$ is the inertia tensor obtained by translating frame L_k to the center of mass of link k and then applying Eq. (6-2-2). Since L_k is attached to link k and rotates with it, the inertia tensor $\bar{D}_k$ is *constant*. The left side of Eq. (6-2-4) represents the kinetic energy due to angular velocity in base frame coordinates, while the right side of Eq. (6-2-4) represents the same energy term expressed in frame L_k coordinates. Since Eq. (6-2-4) must hold for all values of angular velocity $\bar{\omega}^k$, it follows that:

$$D_k(q) = R_0^k(q) \bar{D}_k [R_0^k(q)]^T \qquad (6\text{-}2\text{-}5)$$

Thus the dependence of D_k on q arises through the rotation matrix $R_0^k(q)$ in Eq. (6-2-3). Note that in Eq. (6-2-5), $(R_0^k)^T = R_k^0$ transforms from frame L_0 to frame L_k and then R_0^k transforms from frame L_k back to frame L_0.

To compute the link inertia tensor D_k, we first compute the inertia tensor with respect to a coordinate frame obtained by translating frame L_k from the end of link k to the center of mass of link k. If the D-H algorithm is applied with some care, frame L_k can often be chosen in such a way that $\bar{D}_k$ is diagonal. Whether it is diagonal or not, it is constant, and once this constant matrix $\bar{D}_k$ is obtained, the kth-link inertia tensor with respect to the base frame can then be obtained from it using Eq. (6-2-5).

6-2-2 Link Jacobian

To develop an explicit formulation of the total kinetic energy of the arm, we must also express the velocities $\bar{v}^k$ and $\bar{\omega}^k$ in Eq. (6-2-1) in terms of q and $\dot{q}$. Fortunately, much of the work for doing this is already in place. Recall from the static analysis in Chap. 5 that the manipulator Jacobian matrix $J(q)$ relates infinitesimal displacements of the joint variables to infinitesimal linear and angular displacements of the tool. If we divide both sides of Eq. (5-7-2) by dt and recall Eq. (5-7-1), we see that:

$$\begin{bmatrix} \dot{p} \\ \dot{\phi} \end{bmatrix} = J(q)\dot{q} \qquad (6\text{-}2\text{-}6)$$

Thus $J(q)$ maps instantaneous joint velocity into instantaneous linear and angular tool velocity. To express $\bar{v}^k$ and $\bar{\omega}^k$ in terms of q and $\dot{q}$, we need to find an analogous *link Jacobian matrix*, $J^k(q)$. We can reformulate $J(q)$ as $J^k(q)$ by regarding the center

of mass of link k as if it were the tool tip. Once $J^k(q)$ is determined in this manner, then:

$$\begin{bmatrix} \bar{v}^k \\ \bar{\omega}^k \end{bmatrix} = J^k(q)\dot{q} \qquad 1 \leq k \leq n \tag{6-2-7}$$

Let $\bar{c}^k$ denote the center of mass of link k in base coordinates as shown in Fig. 6-1. Recall that the D-H algorithm assigns frame $L_k = \{x^k, y^k, z^k\}$ to the end of link k. Next let Δc^k be the homogeneous coordinates of the center of mass of link k expressed with respect to frame L_k rather than frame L_0. That is:

$$\Delta c^k = [\bar{c}^k]^k \qquad 1 \leq k \leq n \tag{6-2-8}$$

Since L_k is attached to the end of link k, it follows that Δc^k is a *constant* displacement vector that depends on the physical size and shape of link k. The position of the center of mass of link k can then be expressed in base coordinates by transforming from frame L_k coordinates to frame L_0 coordinates using the composite homogeneous coordinate transformation matrix:

$$\bar{c}^k(q) = H_1 T_0^k(q)\, \Delta c^k \qquad 1 \leq k \leq n \tag{6-2-9}$$

Here we have used the 3×4 homogeneous coordinate conversion matrix H_1 defined in Eq. (2-4-1) to convert from four-dimensional homogeneous coordinates to three-dimensional physical coordinates. Once the location of the center of mass of link k is known, the Jacobian matrix with respect to this point of reference can then be formulated in a manner similar to the method used in Chap. 5 where the point of reference was the tool tip. In this case the Jacobian matrix for link k is the following $6 \times n$ matrix:

$$J^k(q) = \left[\begin{array}{ccc:c} \dfrac{\partial \bar{c}^k}{\partial q_1} & \cdots & \dfrac{\partial \bar{c}^k}{\partial q_k} & 0 \\ \hdashline \xi_1 z^0 & \cdots & \xi_k z^{k-1} & 0 \end{array}\right] \triangleq \left[\begin{array}{c} A^k(q) \\ \hdashline B^k(q) \end{array}\right] \tag{6-2-10}$$

Recall that the scalar ξ_i is a *joint type* parameter that takes on a value of 1 if joint i is revolute or 0 if joint i is prismatic. The vector z^i represents the third unit vector of frame L_i with respect to the base frame. Therefore:

$$z^i(q) = R_0^i(q) i^3 \qquad 0 \leq i \leq n \tag{6-2-11}$$

Note that the last $n - k$ columns of $J^k(q)$ are zero. This is because the motion of link k is not affected by any of the distal joint variables $\{q_{k+1}, \ldots, q_n\}$.

6-2-3 Manipulator Inertia Tensor

An expression for the total kinetic energy of the arm can now be written explicitly in terms of q and $\dot{q}$. First note from Eq. (6-2-10) that if we use the partitioned submatrices A^k and B^k, then Eq. (6-2-7) can be decomposed into two separate equations:

$$\bar{v}^k(q, \dot{q}) = A^k(q)\dot{q} \qquad 1 \leq k \leq n \tag{6-2-12}$$

$$\bar{\omega}^k(q, \dot{q}) = B^k(q)\dot{q} \qquad 1 \leq k \leq n \tag{6-2-13}$$

The first equation specifies the linear velocity of the center of mass of link k, while the second specifies the angular velocity about the center of mass. If we now substitute Eqs. (6-2-12) and (6-2-13) in the original expression for the total kinetic energy in Eq. (6-2-1), the result is:

$$\begin{aligned} T(q, \dot{q}) &= \sum_{k=1}^{n} \frac{(\bar{v}^k)^T m_k \bar{v}^k + (\bar{\omega}^k)^T D_k \bar{\omega}^k}{2} \\ &= \sum_{k=1}^{n} \frac{(A^k\dot{q})^T m_k A^k\dot{q} + (B^k\dot{q})^T D_k B^k\dot{q}}{2} \\ &= \sum_{k=1}^{n} \frac{\dot{q}^T(A^k)^T m_k A^k\dot{q} + \dot{q}^T(B^k)^T D_k B^k\dot{q}}{2} \\ &= \sum_{k=1}^{n} \frac{\dot{q}^T[(A^k)^T m_k A^k + (B^k)^T D_k B^k]\dot{q}}{2} \\ &= \dot{q}^T \frac{\sum_{k=1}^{n} [(A^k)^T m_k A^k + (B^k)^T D_k B^k]}{2} \dot{q} \end{aligned} \tag{6-2-14}$$

Note that the $3 \times n$ Jacobian submatrices A^k and B^k and the 3×3 link inertia tensor D_k depend on q. The expression for the total kinetic energy of the arm can be simplified if we introduce the following $n \times n$ matrix:

$$D(q) \triangleq \sum_{k=1}^{n} \{[A^k(q)]^T m_k A^k(q) + [B^k(q)]^T D_k(q) B^k(q)\} \tag{6-2-15}$$

We call $D(q)$ the *manipulator inertia tensor*. Like the n individual link inertia tensors, the manipulator inertia tensor is a symmetric, positive-definite matrix. Substituting Eq. (6-2-15) in Eq. (6-2-14), we see that the total kinetic energy of the arm can be expressed in terms of the manipulator inertia tensor and the joint velocity as follows:

$$T(q, \dot{q}) = \frac{\dot{q}^T D(q)\dot{q}}{2} \tag{6-2-16}$$

It is clear that the kinetic energy satisfies $T(q, \dot{q}) \geq 0$. Furthermore, $T(q, \dot{q}) = 0$ if and only if $\dot{q} = 0$, which corresponds to the arm being motionless.

6-2-4 Gravity

In order to complete the formulation of the Lagrangian function of a robotic manipulator, it remains to examine the total potential energy of the arm, $U(q)$. The potential energy stored in the kth link of the arm is the amount of work required to displace the center of mass of link k from a horizontal reference plane in the presence of gravity. Let $g \in \mathbf{R}^3$ denote the *gravitational acceleration* vector with respect to the base frame, which is an inertial frame. Then in base frame coordinates, the work

required to displace link k to position $\bar{c}^k(q)$ is $-m_k g^T \bar{c}^k(q)$. Hence the total potential energy stored in the arm is:

$$U(q) = -\sum_{k=1}^{n} m_k g^T \bar{c}^k(q) \tag{6-2-17}$$

If the robot is upright and located near the surface of the earth, then the gravitational acceleration vector is proportional to $-i^3$, where the constant of proportionality is:

$$g_0 = \| g \| = 9.8062 \text{ m/sec}^2 \tag{6-2-18}$$

As was the case with the kinetic energy of the arm, we can simplify the final expression for the potential energy by introducing a new quantity defined as follows:

$$\bar{c}\,(q) \triangleq \sum_{k=1}^{n} m_k \bar{c}^k(q) \tag{6-2-19}$$

Thus $\bar{c}\,(q)$ is a weighted sum of the centers of mass of the n links, with the weighting coefficients being the link masses. From Eq. (6-2-17) we see that the coefficient g^T can be factored out on the left. Thus, combining Eqs. (6-2-17) and (6-2-19), the total potential energy of the arm can be expressed in terms of the weighted sum $\bar{c}\,(q)$ as:

$$U(q) = -g^T \bar{c}\,(q) \tag{6-2-20}$$

Note that $U(q)$ is maximum when all links of the arm are directly opposing g, zero when all links are orthogonal to g, and minimum when all links are aligned with g. A constant could be added to the expression for $U(q)$ to make the minimum potential energy zero. However, this is not really necessary, because from Eq. (6-1-2) we see that it is only the *derivative* of $U(q)$ that is important.

Simplified expressions for the kinetic and potential energy of the arm are now available, and we can take their difference to produce the Lagrangian function of a robotic arm. Combining Eqs. (6-1-1), (6-2-16), and (6-2-20), the manipulator Lagrangian function is:

$$L(q, \dot{q}) = \frac{\dot{q}^T D(q) \dot{q}}{2} + g^T \bar{c}\,(q) \tag{6-2-21}$$

It remains to take derivatives of $L(q, \dot{q})$ with respect to q, $\dot{q}$, and t to formulate the dynamic equations of motion of the arm. However, before doing this, we first develop a more detailed formulation of the term appearing on the right-hand side of Lagrange's equation, the generalized force.

6-3 GENERALIZED FORCE

Generalized forces are the residual forces acting on the arm once the forces associated with kinetic energy (inertial forces) and potential energy (gravitational forces) have been removed. Generalized forces can be defined by examining the work

produced by a virtual displacement of the joints. A *virtual displacement* is an instantaneous infinitesimal displacement δq. A virtual displacement must satisfy the geometric constraints imposed by the linkages of the arm, but it does not have to satisfy temporal constraints, because it is assumed to occur without passage of time. The work produced by a virtual displacement, *virtual work,* is denoted δW. The force that relates a virtual displacement to the virtual work it produces is generalized force. Thus the *generalized force* F, is an $n \times 1$ vector defined implicitly by the following equation:

$$\delta W = F^T \delta q \tag{6-3-1}$$

Note that, depending on the types of joints, the components of the generalized force vector will be either forces (for prismatic joints) or torques (for revolute joints). We investigate two contributions to the generalized force.

6-3-1 Actuators

Each joint is driven or powered by an actuator such as an electric motor or a hydraulic or pneumatic cylinder. Let $\tau \in \mathbf{R}^n$ denote the *joint torque* vector generated by the n actuators. For convenience, τ is referred to as a "torque," but it is understood that τ_k represents a torque if joint k is revolute and a force if joint k is prismatic. Note that τ_k denotes the torque or force actually delivered to axis k. Therefore the effects of actuator dynamics, gears, and mechanical power transmission devices are implicit in τ. The virtual work produced by the actuator torque is:

$$\delta W_1 = \tau^T \delta q \tag{6-3-2}$$

Comparing Eq. (6-3-2) with Eq. (6-3-1), we see that $F = \tau$. Thus the generalized force is simply the actuator torque in this case. However, there is a second source of generalized force to consider.

6-3-2 Friction

The effects of friction can also be modeled as a generalized force applied to the joints of the arm. Friction is a complex nonlinear force that is difficult to model accurately, yet in many cases it can have a significant effect on robot arm dynamics. We examine three distinct components of friction using the followng frictional force model for joint k:

$$b_k(\dot{q}) = b_k^v \dot{q}_k + \operatorname{sgn}(\dot{q}_k)\left[b_k^d + (b_k^s - b_k^d)\exp\frac{-|\dot{q}_k|}{\epsilon}\right] \qquad 1 \le k \le n \tag{6-3-3}$$

Here sgn denotes the signum or sign function. The first term in Eq. (6-3-3) represents *viscous friction* where b_k^v is the coefficient of viscous friction for joint k. The second term in Eq. (6-3-3) represents *dynamic friction* where b_k^d is the coefficient of dynamic friction for joint k. Finally, the third term in Eq. (6-3-3) represents *static friction* or Coulomb friction where b_k^s is the coefficient of static friction for joint k and ϵ is a small positive parameter. Observe from Eq. (6-3-3) that as the velocity of joint k approaches zero, the frictional force approaches $\pm b_k^s$. Consequently, b_k^s can

be interpreted as the torque or force required to overcome friction when motion of joint k is initiated. It is clear that viscous friction is a linear function of $\dot{q}_k$ whereas dynamic and static friction are discontinuous nonlinear functions. The model of frictional forces acting on the kth joint is summarized in graphical form in Fig. 6-3.

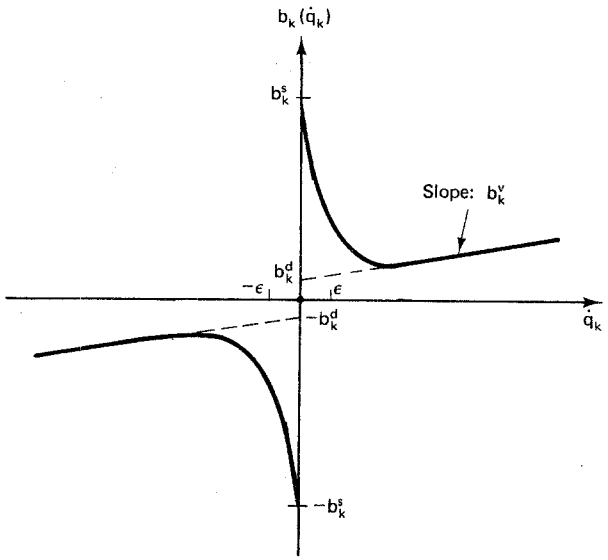


Figure 6-3 Friction model for joint k.

For a well-designed robotic arm, particularly a direct-drive arm, the coefficients of friction will be small. Note that Eq. (6-3-3) is only a rough model of the frictional forces. A more comprehensive representation of friction might employ a term of the form $b(q, \dot{q})$ which depends on joint position as well as joint velocity. This would be necessary, for example, if the shafts about which the links rotate or translate were not perfectly round and uniform. The virtual work produced by the robot due to the frictional forces opposing the motion is:

$$\delta W_2 = -b(\dot{q})^T \, \delta q \tag{6-3-4}$$

The two contributions to the generalized force can now be combined. From Eqs. (6-3-2) and (6-3-4), the total virtual work produced by the actuators and friction is:

$$\delta W = \delta W_1 + \delta W_2 = [\tau - b(\dot{q})]^T \, \delta q \tag{6-3-5}$$

If we now compare Eq. (6-3-5) with Eq. (6-3-1), we see that the expression for the generalized force applied to a robotic arm is:

$$F = \tau - b(\dot{q}) \tag{6-3-6}$$

The friction term in Eq. (6-3-6) has a negative sign because frictional forces *oppose* the motion generated by the actuators. It is also possible to add a third term to the generalized force vector, the joint torques induced by an end-of-arm force and moment vector F^{tool}. However, to keep the dynamic model simple, we will assume

that the arm is moving freely in its workspace and that any payload it is carrying is implicitly included in the physical description of the last link. Under these conditions, the end-of-arm force and moment vector is $F^{\text{tool}} = 0$.

6-4 LAGRANGE-EULER DYNAMIC MODEL

Now that detailed expressions for kinetic energy, potential energy, and generalized force are available, we can apply Lagrange's equation in Eq. (6-1-2) to develop a general dynamic model of a robotic arm. To facilitate computing the required derivatives, we expand the formula for kinetic energy in Eq. (6-2-16) and that for potential energy in Eq. (6-2-20) into sums of scalars, as follows:

$$T(q, \dot{q}) = \frac{\sum_{k=1}^{n} \sum_{j=1}^{n} D_{kj}(q)\dot{q}_k\dot{q}_j}{2} \tag{6-4-1}$$

$$U(q) = -\sum_{k=1}^{3} \sum_{j=1}^{n} g_k m_j \bar{c}_k^j(q) \tag{6-4-2}$$

Recall from Eq. (6-1-1) that the Lagrangian is $L(q, \dot{q}) = T(q, \dot{q}) - U(q)$. Since $D(q)$ is symmetric, the derivative of the Lagrangian with respect to $\dot{q}_i$ can be formulated as:

$$\begin{aligned}
\frac{\partial}{\partial \dot{q}_i} L(q, \dot{q}) &= \frac{\partial}{\partial \dot{q}_i} T(q, \dot{q}) \\
&= \frac{\partial}{\partial q_i} \frac{\sum_{k=1}^{n} \sum_{j=1}^{n} D_{kj}(q)\dot{q}_k\dot{q}_j}{2} \\
&= \frac{\partial}{\partial q_i} \frac{D_{ii}(q)\dot{q}_i^2 + \dot{q}_i\left[\sum_{j\neq i} D_{ij}(q)\dot{q}_j\right] + \dot{q}_i\left[\sum_{k\neq i} D_{ki}(q)\dot{q}_k\right]}{2} \\
&= \frac{2D_{ii}(q)\dot{q}_i + \sum_{j\neq i} D_{ij}(q)\dot{q}_j + \sum_{k\neq i} D_{ki}(q)\dot{q}_k}{2} \\
&= D_{ii}(q)\dot{q}_i + \frac{\sum_{j\neq i} D_{ij}(q)\dot{q}_j + \sum_{k\neq i} D_{ik}(q)\dot{q}_k}{2} \\
&= \sum_{j=1}^{n} D_{ij}(q)\dot{q}_j
\end{aligned} \tag{6-4-3}$$

It then follows that the first term in Lagrange's equation for a robotic arm is:

$$\frac{d}{dt}\frac{\partial}{\partial \dot{q}_i}L(q,\dot{q}) = \frac{d}{dt}\sum_{j=1}^{n} D_{ij}(q)\dot{q}_j$$

$$= \sum_{j=1}^{n} D_{ij}(q)\ddot{q}_j + \sum_{j=1}^{n}\left(\frac{dD_{ij}(q)}{dt}\right)\dot{q}_j$$

$$= \sum_{j=1}^{n} D_{ij}(q)\ddot{q}_j + \sum_{j=1}^{n}\left\{\sum_{k=1}^{n}\left[\frac{\partial D_{ij}(q)}{\partial q_k}\right]\dot{q}_k\right\}\dot{q}_j$$

$$= \sum_{j=1}^{n} D_{ij}(q)\ddot{q}_j + \sum_{k=1}^{n}\sum_{j=1}^{n}\left[\frac{\partial D_{ij}(q)}{\partial q_k}\right]\dot{q}_k\dot{q}_j \qquad (6\text{-}4\text{-}4)$$

Next we examine the second term in Eq. (6-1-2), which is the derivative of the Lagrangian function with respect to q_i. Using Eq. (6-2-12), we have:

$$\frac{\partial}{\partial q_i}L(q,\dot{q}) = \frac{\partial}{\partial q_i}T(q,\dot{q}) - \frac{\partial}{\partial q_i}U(q)$$

$$= \frac{\partial}{\partial q_i}\frac{\sum_{k=1}^{n}\sum_{j=1}^{n} D_{kj}(q)\dot{q}_k\dot{q}_j}{2} + \frac{\partial}{\partial q_i}\sum_{k=1}^{3}\sum_{j=1}^{n} g_k m_j \bar{c}_k^j(q)$$

$$= \frac{\sum_{k=1}^{n}\sum_{j=1}^{n}[\partial D_{kj}(q)/\partial q_i]\dot{q}_k\dot{q}_j}{2} + \sum_{k=1}^{3}\sum_{j=1}^{n} g_k m_j\left[\frac{\partial \bar{c}_k^j(q)}{\partial q_i}\right]$$

$$= \frac{\sum_{k=1}^{n}\sum_{j=1}^{n}[\partial D_{kj}(q)/\partial q_i]\dot{q}_k\dot{q}_j}{2} + \sum_{k=1}^{3}\sum_{j=1}^{n} g_k m_j A_{ki}^j(q)$$

$$= \frac{\sum_{k=1}^{n}\sum_{j=1}^{n}[\partial D_{kj}(q)/\partial q_i]\dot{q}_k\dot{q}_j}{2} + \sum_{k=1}^{3}\sum_{j=i}^{n} g_k m_j A_{ki}^j(q) \qquad (6\text{-}4\text{-}5)$$

The index on the last summation goes from $j = i$ rather than $j = 1$ because, from Eq. (6-2-10), the last $n - j$ columns of A^j are zero. To simplify the final equations of motion, we introduce two new quantities:

$$C_{kj}^i(q) \triangleq \frac{\partial}{\partial q_k}D_{ij}(q) - \frac{1}{2}\frac{\partial}{\partial q_i}D_{kj}(q) \qquad 1 \le i, j, k \le n \qquad (6\text{-}4\text{-}6)$$

$$h_i(q) \triangleq -\sum_{k=1}^{3}\sum_{j=i}^{n} g_k m_j A_{ki}^j(q) \qquad 1 \le i \le n \qquad (6\text{-}4\text{-}7)$$

We refer to the $n \times n$ matrix C^i as the *velocity coupling* matrix for joint i, and the $n \times 1$ vector h as the *gravity loading* vector. With the aid of C^i and h, we can now develop a concise formulation of a dynamic model of a robotic arm.

Proposition 6-4-1: Lagrange-Euler Equations. Let q denote the joint variables and let τ denote the actuator torques of an n-axis robotic arm. If the manipulator is moving freely in its workspace, then its dynamic equations of motion are:

$$\sum_{j=1}^{n} D_{ij}(q)\ddot{q}_j + \sum_{k=1}^{n}\sum_{j=1}^{n} C_{kj}^{i}(q)\dot{q}_k\dot{q}_j + h_i(q) + b_i(\dot{q}) = \tau_i \qquad 1 \le i \le n$$

Proof. Let $1 \le i \le n$. Then, substituting Eqs. (6-3-6), (6-4-4), and (6-4-5) in Eq. (6-1-2) and using Eqs. (6-4-6) and (6-4-7), we have:

$$\begin{aligned}
\tau_i &= F_i + b_i(\dot{q}) \\
&= \frac{d}{dt}\frac{\partial}{\partial \dot{q}_i}L(q, \dot{q}) - \frac{\partial}{\partial q_i}L(q, \dot{q}) + b_i(\dot{q}) \\
&= \sum_{j=1}^{n} D_{ij}(q)\ddot{q}_j + \sum_{k=1}^{n}\sum_{j=1}^{n}\left[\frac{\partial D_{ij}(q)}{\partial q_k}\right]\dot{q}_k\dot{q}_j - \frac{\partial}{\partial q_i}L(q, \dot{q}) + b_i(\dot{q}) \\
&= \sum_{j=1}^{n} D_{ij}(q)\ddot{q}_j + \sum_{k=1}^{n}\sum_{j=1}^{n}\left[\frac{\partial D_{ij}(q)}{\partial q_k} - \frac{\partial D_{kj}(q)/\partial q_i}{2}\right]\dot{q}_k\dot{q}_j + h_i(q) + b_i(\dot{q}) \\
&= \sum_{j=1}^{n} D_{ij}(q)\ddot{q}_j + \sum_{k=1}^{n}\sum_{j=1}^{n} C_{kj}^{i}(q)\dot{q}_k\dot{q}_j + h_i(q) + b_i(\dot{q})
\end{aligned}$$

The first term in Prop. 6-4-1 is an acceleration term that represents the inertial forces and torques generated by the motion of the links of the arm. The second term is a product velocity term associated with Coriolis and centrifugal forces. The third term is a position term representing loading due to gravity. Finally, the fourth term is a velocity term representing the friction opposing the motion of the arm.

The n separate scalar equations in Prop. 6-4-1 can be combined into a single vector equation that represents a dynamic model of the arm. To facilitate this, consider the following quadratic function of the joint velocity:

$$c_i(q, \dot{q}) \triangleq \dot{q}^T C^i(q)\dot{q} \qquad 1 \le i \le n \tag{6-4-8}$$

Note that $c_i(q, \dot{q})$ is the product velocity term in Prop. 6-4-1. Hence the n separate scalar equations in Prop. 6-4-1 can be written as a single vector equation, as follows:

$$D(q)\ddot{q} + c(q, \dot{q}) + h(q) + b(\dot{q}) = \tau \tag{6-4-9}$$

The vector $c(q, \dot{q})$ is called the *velocity coupling* vector. It is clear from Eq. (6-4-8) that $c(q, 0) = 0$. There are two distinct types of interaxis velocity coupling that arise. To see this, we expand the velocity coupling vector as follows:

$$c_i(q, \dot{q}) = \sum_{k=1}^{n} C_{kk}^{i}(q)\dot{q}_k^2 + \sum_{k=1}^{n}\sum_{j\neq k} C_{kj}^{i}(q)\dot{q}_k\dot{q}_j \tag{6-4-10}$$

The two summations in Eq. (6-4-10) are generated by the diagonal and off-diagonal components of the velocity coupling matrices, respectively. The first sum-

mation corresponds to squared velocity terms associated with *centrifugal* forces. For example, the term $C_{kk}^{i}(q)\dot{q}_k^2$ represents the centrifugal force acting on joint i due to the motion of joint k.

The second summation in Eq. (6-4-10) corresponds to product velocity terms associated with *Coriolis* forces. In this case the term $C_{kj}^{i}(q)\dot{q}_k\dot{q}_j$ represents the Coriolis force acting on joint i due to the combined motions of joints k and j where $j \neq k$.

Exercise 6-4-1: One-Axis Robot. Show that the velocity coupling vector of a one-axis robot is always $c(q, \dot{q}) = 0$.

The derivation of the closed-form equations of motion of a robotic arm based on the Lagrange-Euler formulation can be summarized in the form of an algorithm.

Algorithm 6-4-1: Lagrange-Euler Equations

1. Apply the D-H algorithm to assign link coordinates $\{L_0, L_1, \ldots, L_n\}$ orienting frame L_i with the principal axes of link i.
2. Set $T_0^0 = I$, $i = 1$, $D(q) = 0$.
3. Find Δc^i, the homogeneous coordinates of the center of mass of link i with respect to frame L_i.
4. Let L_c be the translation of frame L_i to the center of mass of link i. Use Appendix 2 or Eq. (6-2-2) to compute $\overline{D}_i$, the inertia tensor of link i about its center of mass expressed with respect to frame L_c.
5. Compute:

$$z^{i-1}(q) = R_0^{i-1}(q)i^3$$

$$T_0^i(q) = T_0^{i-1}(q)T_{i-1}^i(q)$$

$$\overline{c}^i(q) = H_1 T_0^i(q)\,\Delta c^i$$

$$D_i(q) = R_0^i(q)\overline{D}_i[R_0^i(q)]^T$$

6. Compute $J^i(q)$ using Eq. (6-2-10).
7. Partition $J^i(q)$ as in Eq. (6-2-10) and compute:

$$D(q) = D(q) + [A^k(q)]^T m_k A^k(q) + [B^k(q)]^T D_k(q) B^k(q)$$

8. Set $i = i + 1$. If $i \leq n$, go to step 3; else, set $i = 1$ and continue.
9. Compute $C^i(q)$, $h_i(q)$, and $b_i(\dot{q})$ using Eqs. (6-4-6), (6-4-7), and (6-3-3), respectively.
10. Formulate the ith equation as:

$$\sum_{j=1}^{n} D_{ij}(q)\ddot{q}_j + \sum_{k=1}^{n}\sum_{j=1}^{n} C_{kj}^{i}(q)\dot{q}_k\dot{q}_j + h_i(q) + b_i(\dot{q}) = \tau_i$$

11. Set $i = i + 1$. If $i \leq n$, go to step 9.

Note that Algorithm 6-4-1 is a two-pass algorithm in terms of the manipulator. Initialization is performed in steps 1 and 2. Pass 1 then consists of steps 3 through 7 repeated for each link, proceeding from link 1 through link n. At the end of pass 1, the manipulator inertia tensor $D(q)$ and the link Jacobian matrices $\{J^k(q)\}$ are available. These quantities serve as inputs to the second pass through the links of the manipulator in steps 9 and 10.

6-5 DYNAMIC MODEL OF A TWO-AXIS PLANAR ARTICULATED ROBOT

A complete dynamic model of an n-axis robotic manipulator can be rather complex. To illustrate the derivation of a complete dynamic model, we first examine a simple two-axis planar articulated robotic arm as shown in Fig. 6-4.

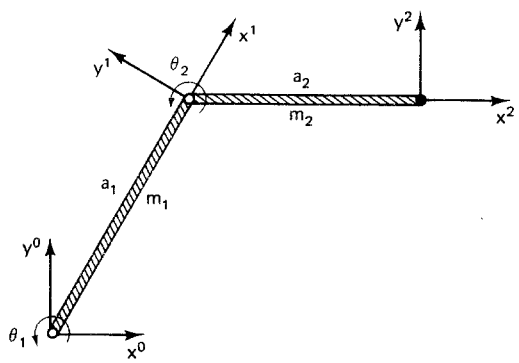


Figure 6-4 A two-axis planar articulated robot.

For convenience, we assume that the two physical links are thin cylinders or rods of mass m_1 and m_2, respectively. We also assume that the mass of any tool attached to the end of link 2 is sufficiently small in comparison with m_2 that it can be ignored.

Applying step 1 of Algorithm 6-4-1 yields the link-coordinate diagram shown in Fig. 6-4. Note that the link-coordinate frames have been assigned in such a manner that the axes of frames L_1 and L_2 are aligned with the principal axes of rods m_1 and m_2, respectively. Since the robot is articulated, the vector of joint variables is $q = \theta$. The kinematic parameters of the two-axis planar articulated robot are listed in Table 6-1.

TABLE 6-1 KINEMATIC PARAMETERS OF THE TWO-AXIS PLANAR ARTICULATED ROBOT

Joint	d	a	θ	α	Home
1	0	a_1	q_1	0	$\pi/3$
2	0	a_2	q_2	0	$-\pi/3$

Next we apply steps 3 through 7 of Algorithm 6-4-1 with $i = 1$. Assuming that the links are homogeneous, we see from Fig. 6-4 that the center of mass of link 1 in frame L_1 coordinates is:

$$\Delta c^1 = \left[-\frac{a_1}{2},\ 0,\ 0,\ 1\right]^T \tag{6-5-1}$$

Let L_c denote the coordinate frame obtained by translating frame L_1 along axis x^1 by an amount $-a_1/2$. L_c is then at the center of mass of link 1, and the axes of L_c are aligned with the principal axes of link 1. If $\overline{D}_1$ is the inertia tensor of link 1 about its center of mass with respect to frame L_c, then, from Appendix 2:

$$\overline{D}_1 = \left(\frac{m_1 a_1^2}{12}\right) \text{diag } \{0,\ 1,\ 1\} \tag{6-5-2}$$

Here the notation diag $\{a, b, c\}$ is short for a diagonal matrix with diagonal elements $\{a, b, c\}$. Next, we apply step 5 where $z^0 = i^3$. Using Table 6-1 and Prop. 2-6-1, or using Eq. (3-7-1) directly, the homogeneous coordinate transformation matrix for link 1 is:

$$T_0^1 = \left[\begin{array}{ccc:c} C_1 & -S_1 & 0 & a_1 C_1 \\ S_1 & C_1 & 0 & a_1 S_1 \\ 0 & 0 & 1 & 0 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{6-5-3}$$

Combining Eqs. (6-5-1) and (6-5-3) and recalling Eq. (2-4-1), the center of mass of link 1 in base frame coordinates is:

$$\begin{aligned} \overline{c}^1 &= H_1 T_0^1 \Delta c^1 \\ &= \left[\frac{a_1 C_1}{2}, \frac{a_1 S_1}{2}, 0\right]^T \end{aligned} \tag{6-5-4}$$

Finally, using Eq. (6-5-2) and the rotation part of T_0^1 in Eq. (6-5-3), the inertia tensor of link 1 in base coordinates is:

$$\begin{aligned} D_1 &= R_0^1 \overline{D}_1 (R_0^1)^T \\ &= \frac{m_1 a_1^2}{12} \begin{bmatrix} S_1^2 & -S_1 C_1 & 0 \\ -S_1 C_1 & C_1^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{6-5-5}$$

Next, the Jacobian matrix for link 1 is computed in step 6. Using Eqs. (6-2-10) and (6-5-4), and recalling that $z^0 = i^3$ and $\xi_1 = 1$, we have the following 6×2 link Jacobian matrix:

$$J^1 = \left[\begin{array}{cc} -a_1 S_1/2 & 0 \\ a_1 C_1/2 & 0 \\ 0 & 0 \\ \hdashline 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{array}\right] \tag{6-5-6}$$

Partitioning J^1 into A^1 and B^1 as in Eq. (6-2-10), the contribution to the manipulator inertia tensor from link 1 is:

$$\begin{aligned} D(q) &= (A^1)^T m_1 A^1 + (B^1)^T D_1 B^1 \\ &= \left(\frac{m_1 a_1^2}{4}\right) \text{diag}\{1, 0\} + \left(\frac{m_1 a_1^2}{12}\right) \text{diag}\{1, 0\} \\ &= \left(\frac{m_1 a_1^2}{3}\right) \text{diag}\{1, 0\} \end{aligned} \tag{6-5-7}$$

This completes the analysis for link 1. Next, we set $i = 2$ and repeat steps 3 through 7. From Fig. 6-4, the center of mass of link 2 in frame L_2 coordinates is:

$$\Delta c^2 = \left[\frac{-a_2}{2}, 0, 0, 1\right]^T \tag{6-5-8}$$

Let L_c denote the coordinate frame obtained by translating frame L_2 along axis x^2 by an amount $-a_2/2$. L_c is then at the center of mass of link 2, and the axes of L_c are aligned with the principal axes of link 2. If $\bar{D}_2$ is the inertia tensor of link 2 about its center of mass with respect to frame L_c, then, from Appendix 2:

$$\bar{D}_2 = \left(\frac{m_2 a_2^2}{12}\right) \text{diag}\{0, 1, 1\} \tag{6-5-9}$$

Next, we apply step 5. From Eq. (6-5-3), extracting the third column of R_0^1 yields $z^1 = R_0^1 i^3 = i^3$. From Table 6-1, Prop. 2-6-1, and Eq. (6-5-3), or from Eq. (3-7-1) directly, the homogeneous coordinate transformation matrix for link 2 is:

$$T_0^2 = \left[\begin{array}{ccc:c} C_{12} & -S_{12} & 0 & a_1 C_1 + a_2 C_{12} \\ S_{12} & C_{12} & 0 & a_1 S_1 + a_2 S_{12} \\ 0 & 0 & 1 & 0 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{6-5-10}$$

Combining Eqs. (6-5-8) and (6-5-10), the center of mass of link 2 in base frame coordinates is:

$$\begin{aligned} \bar{c}^2 &= H_1 T_0^2 \Delta c^2 \\ &= \left[a_1 C_1 + \frac{a_2 C_{12}}{2}, a_1 S_1 + \frac{a_2 S_{12}}{2}, 0\right]^T \end{aligned} \tag{6-5-11}$$

Finally, using Eq. (6-5-9) and the rotation part of T_0^2 in Eq. (6-5-10), the inertia tensor of link 2 in base coordinates is:

$$\begin{aligned} D_2 &= R_0^2 \bar{D}_2 (R_0^2)^T \\ &= \frac{m_2 a_2^2}{12} \begin{bmatrix} S_{12}^2 & -S_{12}C_{12} & 0 \\ -S_{12}C_{12} & C_{12}^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{6-5-12}$$

The Jacobian matrix for link 2 is now computed in step 6. Using Eqs. (6-2-10) and (6-5-11) and recalling that $z^1 = i^3$ and $\xi_2 = 1$, we have the following 6×2 link Jacobian matrix:

$$J^2 = \begin{bmatrix} -a_1 S_1 - a_2 S_{12}/2 & -a_2 S_{12}/2 \\ a_1 C_1 + a_2 C_{12}/2 & a_2 C_{12}/2 \\ 0 & 0 \\ \hline 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \tag{6-5-13}$$

Partitioning J^2 into A^2 and B^2 as in Eq. (6-2-10), we update the expression for the manipulator inertia tensor in Eq. (6-5-7) by adding the contribution from link 2. Since $n = 2$, this generates the complete manipulator inertia tensor of the two-axis planar articulated robot:

$$\begin{aligned} D(q) &= D(q) + (A^2)^T m_2 A^2 + (B^2)^T D_2 B_2 \\ &= D(q) + m_2 \begin{bmatrix} a_1^2 + a_1 a_2 C_2 + \frac{a_2^2}{4} & \frac{a_1 a_2 C_2}{2} + \frac{a_2^2}{4} \\ \frac{a_1 a_2 C_2}{2} + \frac{a_2^2}{4} & \frac{a_2^2}{4} \end{bmatrix} + \frac{m_2 a_2^2}{12} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{m_1 a_1^2}{3} + m_2\left(a_1^2 + a_1 a_2 C_2 + \frac{a_2^2}{3}\right) & m_2\left(\frac{a_1 a_2 C_2}{2} + \frac{a_2^2}{3}\right) \\ m_2\left(\frac{a_1 a_2 C_2}{2} + \frac{a_2^2}{3}\right) & \frac{m_2 a_2^2}{3} \end{bmatrix} \end{aligned} \tag{6-5-14}$$

This completes the first pass of Algorithm 6-4-1. Next we apply steps 9 and 10. Setting $i = 1$ in Eq. (6-4-6) and using Eq. (6-5-14), the velocity coupling matrix for joint 1 is

$$C^1 = \frac{m_2 a_1 a_2 S_2}{2} \begin{bmatrix} 0 & 0 \\ -2 & -1 \end{bmatrix} \tag{6-5-15}$$

If we assume that the y^0 axis in Fig. 6-4 points straight up, directly opposing gravity, then $g = [0, -g_0, 0]^T$, where $g_0 = 9.8062$ m/sec^2. Applying Eq. (6-4-7) and using Eqs. (6-5-6) and (6-5-13), the gravitational loading on joint 1 is:

$$\begin{aligned} h_1 &= g_0(m_1 A_{21}^1 + m_2 A_{21}^2) \\ &= g_0\left[\left(\frac{m_1}{2} + m_2\right) a_1 C_1 + \frac{m_2 a_2 C_{12}}{2}\right] \end{aligned} \tag{6-5-16}$$

Finally, using Eqs. (6-5-14) through (6-5-16), we can formulate the dynamic model for the first joint of the manipulator using step 10, as follows:

$$\tau_1 = \left[\left(\frac{m_1}{3} + m_2\right)a_1^2 + m_2 a_1 a_2 C_2 + \frac{m_2 a_2^2}{3}\right]\ddot{q}_1 + \left(\frac{m_2 a_1 a_2 C_2}{2} + \frac{m_2 a_2^2}{3}\right)\ddot{q}_2$$
$$- m_2 a_1 a_2 S_2\left(\dot{q}_1\dot{q}_2 + \frac{\dot{q}_2^2}{2}\right) + g_0\left[\left(\frac{m_1}{2} + m_2\right)a_1 C_1 + \frac{m_2 a_2 C_{12}}{2}\right] + b_1(\dot{q}_1) \quad (6\text{-}5\text{-}17)$$

The equation for the first joint is typically the most complex, because its inertial and gravitational terms will depend on the remaining links all the way out to the end of the arm. As we proceed toward the end of the arm, the dynamic equations become progressively simpler. We again apply steps 9 and 10 to produce the dynamic equation for joint 2. Setting $i = 2$ in Eq. (6-4-6) and using Eq. (6-5-14), the velocity coupling matrix for joint 2 is:

$$C^2 = \frac{m_2 a_1 a_2 S_2}{4}\begin{bmatrix} 2 & 1 \\ -1 & 0 \end{bmatrix} \quad (6\text{-}5\text{-}18)$$

Applying Eq. (6-4-7) and using Eq. (6-5-13), the gravitational loading on joint 2 is then:

$$h_2 = g_0 m_2 A_{22}^2$$
$$= \frac{g_0 m_2 a_2 C_{12}}{2} \quad (6\text{-}5\text{-}19)$$

Finally, using Eqs. (6-5-14), (6-5-18), and (6-5-19), we can formulate the dynamic model for the second joint of the manipulator using step 10, as follows:

$$\tau_2 = \left(\frac{m_2 a_1 a_2 C_2}{2} + \frac{m_2 a_2^2}{3}\right)\ddot{q}_1 + \frac{m_2 a_2^2 \ddot{q}_2}{3} + \frac{m_2 a_1 a_2 S_2 \dot{q}_1^2}{2}$$
$$+ \frac{g_0 m_2 a_2 C_{12}}{2} + b_2(\dot{q}_2) \quad (6\text{-}5\text{-}20)$$

Clearly, the dynamic equation for joint 2 is simpler than the dynamic equation for joint 1, as expected. However, it is evident that even for this simple two-axis robot, the dynamic model in Eqs. (6-5-17) and (6-5-20) is fairly complex. The robot in Fig. 6-4 demonstrates all of the essential features of robot dynamics, including inertial forces, Coriolis and centrifugal velocity coupling, gravitational effects, and friction.

6-6 DYNAMIC MODEL OF A THREE-AXIS SCARA ROBOT

As a second example of a complete dynamic model of a robotic arm, consider the SCARA robot shown in Fig. 6-5. The first three axes of a SCARA robot position the tool tip, while the fourth axis orients the tool through a roll motion. To keep the dynamic model relatively simple, we assume that the mass of the fourth link and any tool attached to it are sufficiently small in comparison with the masses of the other links that they can be ignored. This is not an unreasonable assumption, because the links of a robotic manipulator tend to become progressively smaller and less massive

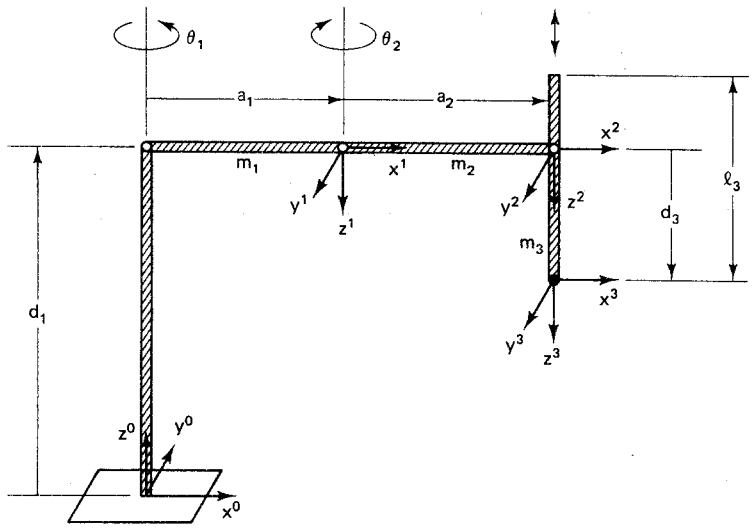


Figure 6-5 A three-axis SCARA robot.

as one proceeds from the base joint to the tool. We assume that the three links are thin cylinders or rods of mass m_1, m_2, and m_3, respectively.

Applying step 1 of Algorithm 6-4-1 yields the link-coordinate diagram shown in Fig. 6-5. Note that the link-coordinate frames have been assigned in such a manner that the axes of frame L_k are aligned with the principal axes of rod m_k for $1 \leq k \leq 3$. In this case the vector of joint variables is $q = [\theta_1, \theta_2, d_3]^T$. The kinematic parameters of the three-axis SCARA robot are listed in Table 6-2.

TABLE 6-2 KINEMATIC PARAMETERS OF THE THREE-AXIS SCARA ROBOT

Joint	d	a	θ	α	Home
1	d_1	a_1	q_1	π	0
2	0	a_2	q_1	0	0
3	q_3	0	0	0	$l_3/2$

Next we apply steps 3 through 7 of Algorithm 6-4-1 with $i = 1$. Note from Fig. 6-5 that the vertical column of height d_1 is stationary; it is only the rod of length a_1 and mass m_1 that rotates when joint 1 is activated. Assuming that the links are homogeneous, we see from Fig. 6-5 that the center of mass of link 1 in frame L_1 coordinates is:

$$\Delta c^1 = \left[-\frac{a_1}{2}, 0, 0, 1 \right]^T \tag{6-6-1}$$

Let L_c denote the coordinate frame obtained by translating frame L_1 along axis x^1 by an amount $-a_1/2$. L_c is then at the center of mass of link 1, and the axes of L_c

are aligned with the principal axes of link 1. If $\overline{D}_1$ is the inertia tensor of link 1 about its center of mass with respect to frame L_c, then, from Appendix 2:

$$\overline{D}_1 = \left(\frac{m_1 a_1^2}{12}\right) \text{diag } \{0, 1, 1\} \tag{6-6-2}$$

Next we apply step 5 where $z^0 = i^3$. Using Table 6-2 and Prop. 2-6-1, or using Eq. (2-8-3) directly, the homogeneous coordinate transformation matrix for link 1 is:

$$T_0^1 = \left[\begin{array}{ccc:c} C_1 & S_1 & 0 & a_1 C_1 \\ S_1 & -C_1 & 0 & a_1 S_1 \\ 0 & 0 & -1 & d_1 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{6-6-3}$$

Combining Eqs. (6-6-1) and (6-6-3) and recalling Eq. (2-4-1), the center of mass of link 1 in base frame coordinates is:

$$\begin{aligned} \overline{c}^1 &= H_1 T_0^1 \, \Delta c^1 \\ &= \left[\frac{a_1 C_1}{2}, \frac{a_1 S_1}{2}, d_1\right]^T \end{aligned} \tag{6-6-4}$$

Finally, using Eq. (6-6-2) and the rotation part of T_0^1 in Eq. (6-6-3), the inertia tensor of link 1 in base coordinates is:

$$\begin{aligned} D_1 &= R_0^1 \overline{D}_1 (R_0^1)^T \\ &= \frac{m_1 a_1^2}{12} \begin{bmatrix} S_1^2 & -S_1 C_1 & 0 \\ -S_1 C_1 & C_1^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{6-6-5}$$

Next, the Jacobian matrix for link 1 is computed in step 6. Using Eqs. (6-2-10) and (6-6-4), and recalling that $z^0 = i^3$ and $\xi_1 = 1$, we have the following 6×3 link Jacobian matrix:

$$J^1 = \left[\begin{array}{ccc} -a_1 S_1/2 & 0 & 0 \\ a_1 C_1/2 & 0 & 0 \\ 0 & 0 & 0 \\ \hdashline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{array}\right] \tag{6-6-6}$$

Partitioning J^1 into A^1 and B^1 as in Eq. (6-2-10), the contribution to the manipulator inertia tensor from link 1 is:

$$\begin{aligned} D(q) &= (A^1)^T m_1 A^1 + (B^1)^T D_1 B^1 \\ &= \left(\frac{m_1 a_1^2}{4}\right) \text{diag } \{1, 0, 0\} + \left(\frac{m_1 a_1^2}{12}\right) \text{diag } \{1, 0, 0\} \end{aligned}$$

$$= \left(\frac{m_1 a_1^2}{3}\right) \text{diag}\{1, 0, 0\} \tag{6-6-7}$$

This completes the analysis for link 1. Next, we set $i = 2$ and repeat steps 3 through 7. From Fig. 6-5, the center of mass of link 2 in frame L_2 coordinates is:

$$\Delta c^2 = \left[-\frac{a_2}{2}, 0, 0, 1\right]^T \tag{6-6-8}$$

Again, let L_c denote the coordinate frame obtained by translating frame L_2 along axis x^2 by an amount $-a_2/2$. L_c is then at the center of mass of link 2, and the axes of L_c are aligned with the principal axes of link 2. If $\overline{D}_2$ is the inertia tensor of link 2 about its center of mass with respect to frame L_c, then, from Appendix 2:

$$\overline{D}_2 = \left(\frac{m_2 a_2^2}{12}\right) \text{diag}\{0, 1, 1\} \tag{6-6-9}$$

Next, we apply step 5. Extracting the third column of R_0^1 in Eq. (6-6-3) yields $z^1 = R_0^1 i^3 = -i^3$. From Table 6-2, Prop. 2-6-1, and Eq. (6-6-3), or from Eq. (2-8-3) directly, the homogeneous coordinate transformation matrix for link 2 is:

$$T_0^2 = \left[\begin{array}{ccc|c} C_{1-2} & S_{1-2} & 0 & a_1 C_1 + a_2 C_{1-2} \\ S_{1-2} & -C_{1-2} & 0 & a_1 S_1 + a_2 S_{1-2} \\ 0 & 0 & -1 & d_1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{6-6-10}$$

Combining Eqs. (6-6-8) and (6-6-10), the center of mass of link 2 in base frame coordinates is:

$$\begin{aligned} \overline{c}^2 &= H_1 T_0^2 \Delta c^2 \\ &= \left[a_1 C_1 + \frac{a_2 C_{1-2}}{2}, a_1 S_1 + \frac{a_2 S_{1-2}}{2}, d_1\right]^T \end{aligned} \tag{6-6-11}$$

Finally, using Eq. (6-6-9) and the rotation part of T_0^2 in Eq. (6-6-10), the inertia tensor of link 2 in base coordinates is:

$$\begin{aligned} D_2 &= R_0^2 \overline{D}_2 (R_0^2)^T \\ &= \frac{m_2 a_2^2}{12} \begin{bmatrix} S_{1-2}^2 & -S_{1-2}C_{1-2} & 0 \\ -S_{1-2}C_{1-2} & C_{1-2}^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{6-6-12}$$

Next, we compute the Jacobian matrix for link 2 in step 6. Using Eqs. (6-2-10) and (6-6-11), and recalling that $z^1 = -i^3$ and $\xi_2 = 1$, we have the following 6×3 link Jacobian matrix:

$$J^2 = \left[\begin{array}{ccc} -a_1S_1 - a_2S_{1-2}/2 & a_2S_{1-2}/2 & 0 \\ a_1C_1 + a_2C_{1-2}/2 & -a_2C_{1-2}/2 & 0 \\ 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & -1 & 0 \end{array}\right] \tag{6-6-13}$$

Partitioning J^2 into A^2 and B^2 as in Eq. (6-2-10), we can add the contribution to the manipulator inertia tensor from link 2 to the previous expression in Eq. (6-6-7):

$$D(q) = D(q) + (A^2)^T m_2 A^2 + (B^2)^T D_2 B^2$$

$$= D(q) + m_2 \begin{bmatrix} a_1^2 + a_1a_2C_2 + \frac{a_2^2}{4} & -\left(\frac{a_1a_2C_2}{2} + \frac{a_2^2}{4}\right) & 0 \\ -\left(\frac{a_1a_2C_2}{2} + \frac{a_2^2}{4}\right) & \frac{a_2^2}{4} & 0 \\ 0 & 0 & 0 \end{bmatrix} + \frac{m_2a_2^2}{12}\begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} \left(\frac{m_1}{3} + m_2\right)a_1^2 + m_2C_2a_1a_2 + \frac{m_2a_2^2}{3} & -\frac{m_2a_1a_2C_2}{2} - \frac{m_2a_2^2}{3} & 0 \\ -\frac{m_2a_1a_2C_2}{2} - \frac{m_2a_2^2}{3} & \frac{m_2a_2^2}{3} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{6-6-14}$$

This completes the analysis for link 2. Next, we set $i = 3$ and repeat steps 3 through 7. Unlike the first two joints, which are revolute, the third joint of the SCARA robot is prismatic. If we assume that the entire rod of length l_3 and mass m_3 slides up and down when joint 3 is activated, then, from Fig. 6-5, the center of mass of link 3 in frame L_3 coordinates is:

$$\Delta c^3 = \left[0, 0, -\frac{l_3}{2}, 1\right]^T \tag{6-6-15}$$

As before, let L_c denote the coordinate frame obtained by translating frame L_3 along axis z^3 by an amount $-l_3/2$. L_c is then at the center of mass of link 3, and the axes of L_c are aligned with the principal axes of link 3. If $\bar{D}_3$ is the inertia tensor of link 3 about its center of mass with respect to frame L_c, then, from Appendix 2:

$$\bar{D}_3 = \left(\frac{m_3l_3^2}{12}\right) \text{diag}\{1, 1, 0\} \tag{6-6-16}$$

Next, we apply step 5. Extracting the third column of R_0^2 in Eq. (6-6-10) yields $z^2 = R_0^2 i^3 = -i^3$. From Table 6-2, Prop. 2-6-1, and Eq. (6-6-10), or from Eq. (2-8-3) directly, the homogeneous coordinate transformation matrix for link 3 is:

$$T_0^3 = \left[\begin{array}{ccc|c} C_{1-2} & S_{1-2} & 0 & a_1C_1 + a_2C_{1-2} \\ S_{1-2} & -C_{1-2} & 0 & a_1S_1 + a_2S_{1-2} \\ 0 & 0 & -1 & d_1 - q_3 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{6-6-17}$$

Combining Eqs. (6-6-15) and (6-6-17), the center of mass of link 3 in base frame coordinates is:

$$\begin{aligned} \bar{c}^3 &= H_1 T_0^3 \Delta c^3 \\ &= \left[a_1C_1 + a_2C_{1-2},\ a_1S_1 + a_2S_{1-2},\ d_1 - q_3 + \frac{l_3}{2}\right]^T \end{aligned} \tag{6-6-18}$$

Finally, using Eq. (6-6-16) and the rotation part of T_0^3 in Eq. (6-6-17), the inertia tensor of link 3 in base coordinates is:

$$\begin{aligned} D_3 &= R_0^3 \bar{D}_3 (R_0^3)^T \\ &= \left(\frac{m_3 l_3^2}{12}\right) \text{diag}\,\{1,\ 1,\ 0\} \end{aligned} \tag{6-6-19}$$

Next, the Jacobian matrix for link 3 is computed in step 6. Since joint 3 is prismatic, $\xi_3 = 0$. Using Eqs. (6-2-10) and (6-6-18), we then have the following 6×3 link Jacobian matrix:

$$J^3 = \left[\begin{array}{ccc} -a_1S_1 - a_2S_{1-2} & a_2S_{1-2} & 0 \\ a_1C_1 + a_2C_{1-2} & -a_2C_{1-2} & 0 \\ 0 & 0 & -1 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & -1 & 0 \end{array}\right] \tag{6-6-20}$$

Partitioning J^3 into A^3 and B^3 as in Eq. (6-2-10), we can add the contribution to the manipulator inertia tensor from link 3 to the expression in Eq. (6-6-14):

$$\begin{aligned} D(q) &= D(q) + (A^3)^T m_3 A^3 + (B^3)^T D_3 B^3 \\ &= D(q) + m_3 \begin{bmatrix} a_1^2 + 2a_1a_2C_2 + a_2^2 & -(a_1a_2C_2 + a_2^2) & 0 \\ -(a_1a_2C_2 + a_2^2) & a_2^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{6-6-21}$$

The final expression for the manipulator inertia tensor of the three-axis SCARA robot is then $D(q) = [d^1(q), d^2(q), d^3(q)]$, where the three columns of $D(q)$ are:

$$d^1(q) = \begin{bmatrix} \left(\frac{m_1}{3} + m_2 + m_3\right)a_1^2 + (m_2 + 2m_3)a_1a_2C_2 + \left(\frac{m_2}{3} + m_3\right)a_2^2 \\ -\left(\frac{m_2}{2} + m_3\right)a_1a_2C_2 - \left(\frac{m_2}{3} + m_3\right)a_2^2 \\ 0 \end{bmatrix} \tag{6-6-22a}$$

$$d^2(q) = \begin{bmatrix} -\left(\frac{m_2}{2} + m_3\right)a_1a_2C_2 - \left(\frac{m_2}{3} + m_3\right)a_2^2 \\ \left(\frac{m_2}{3} + m_3\right)a_2^2 \\ 0 \end{bmatrix} \tag{6-6-22b}$$

$$d^3(q) = [0, 0, m_3]^T \tag{6-6-22c}$$

This completes the first pass of Algorithm 6-4-1. Next, we apply steps 9 and 10. Setting $i = 1$ in Eq. (6-4-6) and using Eq. (6-6-22), the velocity coupling matrix for joint 1 is:

$$C^1 = a_1a_2S_2\begin{bmatrix} 0 & 0 & 0 \\ -(m_2 + 2m_3) & \frac{m_2}{2} + m_3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{6-6-23}$$

If we assume that the z^0 axis in Fig. 6-5 points straight up, directly opposing gravity, then $g = [0, 0, -g_0]^T$, where $g_0 = 9.8062$ m/sec^2. Applying Eq. (6-4-7) and using Eqs. (6-6-6), (6-6-13), and (6-6-20), the gravitational loading on joint 1 is:

$$h_1 = g_0(m_1A_{31}^1 + m_2A_{31}^2 + m_3A_{31}^3) = 0 \tag{6-6-24}$$

This is as it should be, since axis 1 is aligned with the gravitational field. Using Eqs. (6-6-22), (6-6-23), and (6-6-24), we can formulate the dynamic model for the first joint of the manipulator using step 10 as follows:

$$\begin{aligned} \tau_1 = {} & \left[\left(\frac{m_1}{3} + m_2 + m_3\right)a_1^2 + (m_2 + 2m_3)a_1a_2C_2 + \left(\frac{m_2}{3} + m_3\right)a_2^2\right]\ddot{q}_1 \\ & - \left[\left(\frac{m_2}{2} + m_3\right)a_1a_2C_2 + \left(\frac{m_2}{3} + m_3\right)a_2^2\right]\ddot{q}_2^2 + b_1(\dot{q}_1) \\ & - a_1a_2S_2\left[(m_2 + 2m_3)\dot{q}_1\dot{q}_2 - \left(\frac{m_2}{2} + m_3\right)\dot{q}_2^2\right] \end{aligned} \tag{6-6-25}$$

Next, we apply steps 9 and 10 with $i = 2$ to produce the dynamic equation for joint 2. Setting $i = 2$ in Eq. (6-4-6) and using Eq. (6-6-22), the velocity coupling matrix for joint 2 is:

$$C^2 = a_1 a_2 S_2 \begin{bmatrix} \frac{m_2}{2} + m_3 & -\left(\frac{m_2}{4} + \frac{m_3}{2}\right) & 0 \\ \frac{m_2}{4} + \frac{m_3}{2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{6-6-26}$$

Applying Eq. (6-4-7) and using Eqs. (6-6-13) and (6-6-20), the gravitational loading on joint 2 is then:

$$h_2 = g_0 m_2 (A_{32}^2 + A_{32}^3) = 0 \tag{6-6-27}$$

Again, there is no loading due to gravity on joint 2. Finally, using Eqs. (6-6-22), (6-6-26), and (6-6-27), we can formulate the dynamic model for the second joint of the manipulator using step 10, as follows:

$$\tau_2 = -\left[\left(\frac{m_2}{2} + m_3\right) a_1 a_2 C_2 + \left(\frac{m_2}{3} + m_3\right) a_2^2\right] \ddot{q}_1 + \left(\frac{m_2}{3} + m_3\right) a_2^2 \ddot{q}_2 + \left(\frac{m_2}{2} + m_3\right) a_1 a_2 S_2 \dot{q}_1^2 + b_2(\dot{q}_2) \tag{6-6-28}$$

Finally, we apply steps 9 and 10 with $i = 3$ to produce the dynamic equation for joint 3. Setting $i = 3$ in Eq. (6-4-6) and using Eq. (6-6-22), the velocity coupling matrix for joint 3 is:

$$C^3 = 0 \tag{6-6-29}$$

Applying Eq. (6-4-7) and using Eq. (6-6-20), the gravitational loading on joint 3, the prismatic joint, is then:

$$\begin{aligned} h_3 &= g_0 m_3 A_{33}^3 \\ &= -g_0 m_3 \end{aligned} \tag{6-6-30}$$

Finally, using Eqs. (6-6-22), (6-6-29), and (6-6-30), we can formulate the dynamic model for the third joint of the manipulator using step 10, as follows:

$$\tau_3 = m_3 \ddot{q}_3 - g_0 m_3 + b_3(\dot{q}_3) \tag{6-6-31}$$

The dynamic model for the three-axis SCARA robot in Eqs. (6-6-25), (6-6-28), and (6-6-31) is only slightly more complex than the dynamic model for the two-axis planar articulated robot. This is because a SCARA robot is essentially a two-axis articulated robot used to establish the horizontal tool position, plus an orthogonal one-axis prismatic robot used to establish vertical tool position. The minimal dynamic coupling between these two parts of the robot reduces the complexity of the overall equations of motion. It is clear from Eq. (6-6-31) that the motion of the third joint is completely independent of the other two joints. The motion of the first two joints depends on the mass of the third link, which acts as a load, but is otherwise independent of the third joint.

6-7 DIRECT AND INVERSE DYNAMICS

Recall from Chap. 3 that there are two versions of the robot kinematics problem, the direct problem and the inverse problem. There are also two versions of the robot dynamics problem, a direct dynamics problem and an inverse dynamics problem, as shown in Fig. 6-6.

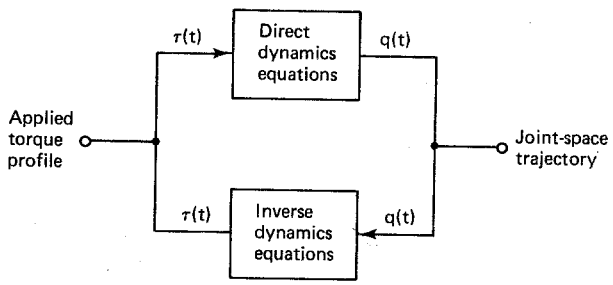


Figure 6-6 Direct and inverse dynamics.

To solve the *direct dynamics problem,* one must compute the joint trajectory $q(t)$ given the applied torque profile $\tau(t)$. Once the joint trajectory is available, it can be differentiated twice to generate the joint velocity $\dot{q}(t)$ and joint acceleration $\ddot{q}(t)$. The joint trajectory can be obtained numerically by solving the nonlinear differential equations in Prop. 6-4-1 using numerical integration techniques (Gear, 1971).

The solution to the *inverse dynamics problem* requires the computation of the applied torque profile $\tau(t)$ corresponding to a given joint trajectory $q(t)$, assuming that $q(t)$ is sufficiently smooth to be twice differentiable. Unlike the inverse kinematics problem, the inverse dynamics problem is actually *simpler* than the direct dynamics problem. Indeed, given a trajectory $q(t)$, we can differentiate it twice and then substitute the results into Eq. (6-4-9) to generate the following expression for the corresponding applied torque:

$$\tau(t) = D(q(t))\ddot{q}(t) + c(q(t), \dot{q}(t)) + h(q(t)) + b(\dot{q}(t)) \qquad (6\text{-}7\text{-}1)$$

The solution to the inverse dynamics problem based on the Lagrange-Euler equations is appealing in the sense that it is a closed-form expression that reveals the *structure* of the solution. The terms in Eq. (6-7-1) each have a direct physical interpretation. The first term, $D(q)\ddot{q}$, represents inertia associated with the distribution of mass. The second term, $c(q, \dot{q})$, represents interaxis velocity coupling due to centrifugal and Coriolis forces. The third term, $h(q)$, represents loading due to gravity, and the last term, $b(\dot{q})$, represents the effects of friction.

Solution of the inverse dynamics problem is useful for the design of robot arm controllers, controllers which exploit a detailed model of the robotic manipulator. Here the objective is to find an applied torque input $\tau(t)$ that will drive the robotic manipulator along a nominal joint trajectory $q(t)$ in joint space. There is one major drawback to using Eq. (6-7-1) as part of a robot controller. It is the loss of speed caused by the substantial computation required. To obtain reasonable tracking performance from a feedback control system, the control signal must be updated at a

rate of at least 100–200 Hz. Consequently, there are about 5 to 10 milliseconds (msec) available to compute new values for $\tau(t)$ as the robot is driven along its trajectory. This is often not enough time to evaluate the complete Lagrange-Euler model. Indeed, from Eqs. (6-2-18) and (6-4-6) and Prop. 4-6-1 we see that the computational complexity of the Lagrange-Euler formulation is of order $O(n^4)$, where n is the number of axes. Here a computation $f(n)$ is of order $O(n^i)$ if the number of basic operations grows as the ith power of n. That is, $f(n)$ is of order $O(n^i)$ if there exists some constant $\gamma \neq 0$ such that:

$$\lim_{n\to\infty} \frac{f(n)}{n^i} = \gamma \tag{6-7-2}$$

Since of the number of basic computations (additions, multiplications) is $O(n^4)$, the computational work increases very rapidly as n increases. Indeed, when $n = 6$, the evaluation of the complete Lagrange-Euler model can require up to 66,271 multiplications and 51,548 additions (Hollerbach, 1980).

6-8 RECURSIVE NEWTON-EULER FORMULATION

A more efficient solution to the inverse dynamics problem can be obtained by using a *recursive* formulation of robot arm dynamics based on the Newton-Euler equations (Fu et al., 1987). In the recursive formulation, the chainlike physical structure of the manipulator is exploited, with the motion of each link represented with respect to its neighboring link. The Newton-Euler formulation makes two passes over the links of the manipulator as shown in Fig. 6-7.

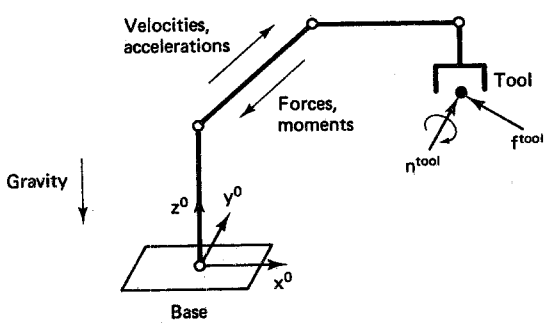


Figure 6-7 Two-pass recursive Newton-Euler formulation.

Given a joint-space trajectory, the velocities and accelerations of each link are computed recursively, starting at the base and propagating forward to the tool. These are called the *forward equations*. Once the link velocities and accelerations are known, this information is then used to compute the forces and moments acting on each link starting at the tool and working backward to the base. These are the *backward equations*.

6-8-1 Forward Newton-Euler Equations

First we formulate the forward equations, which compute the link velocities and accelerations starting at the base and working forward to the tool as illustrated in Fig. 6-8. Unless specifically noted otherwise, *all* quantities are expressed in base frame coordinates. Let $q(t)$ denote the desired joint-space trajectory, and let $\dot{q}(t)$ and $\ddot{q}(t)$ denote the corresponding joint velocity and joint acceleration, respectively.

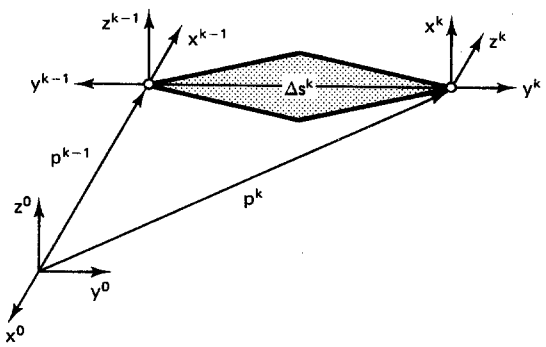


Figure 6-8 Forward Newton-Euler equations.

Rather than write separate equations for revolute joints and prismatic joints, we employ the joint type parameter from Eq. (2-6-2), which is defined as $\xi_k = 1$ when joint k is revolute and $\xi_k = 0$ when joint k is prismatic. Let ω^k denote the *angular velocity* of frame L_k relative to frame L_0. Then ω^k can be expressed recursively as follows:

$$\omega^k = \omega^{k-1} + \xi_k \dot{q}_k z^{k-1} \tag{6-8-1}$$

Note that if joint k is prismatic, the angular velocity of link k is the same as the angular velocity of link $k-1$. Next, let $\dot{\omega}^k$ denote the *angular acceleration* of frame L_k relative to frame L_0. To generate a forward recursive equation for the angular acceleration, we differentiate Eq. (6-8-1). Applying the chain rule, the result can be expressed in terms of the vector cross-product as follows (Fu et al., 1987):

$$\dot{\omega}^k = \dot{\omega}^{k-1} + \xi_k[\ddot{q}_k z^{k-1} + \omega^{k-1} \times (\dot{q}_k z^{k-1})] \tag{6-8-2}$$

Next, let v^k denote the *linear velocity* of frame L_k relative to frame L_0. To concisely formulate the linear velocity, it is helpful to first introduce the following variable shown in Fig. 6-8:

$$\Delta s^k \triangleq p^k - p^{k-1} \tag{6-8-3}$$

Recall that p^k denotes the location of the origin of frame L_k with respect to frame L_0. Thus p^k is the 3×1 vector extracted from the fourth column of the homogeneous coordinate transformation matrix T_0^k. The linear velocity of frame L_k relative to frame L_0 can be expressed recursively in terms of Δs^k as follows (Fu et al., 1987):

$$v^k = v^{k-1} + \omega^k \times \Delta s^k + (1 - \xi_k)\dot{q}_k z^{k-1} \tag{6-8-4}$$

Finally, let $\dot{v}^k$ denote the *linear acceleration* of frame L_k relative to frame L_0. To generate a forward recursive equation for the linear acceleration, we differentiate Eq. (6-8-4). Applying the chain rule and using certain cross-product identities, the final result can be expressed as follows (Fu et al., 1987):

$$\dot{v}^k = \dot{v}^{k-1} + \dot{\omega}^k \times \Delta s^k + \omega^k \times (\omega^k \times \Delta s^k) + (1 - \xi_k)[\ddot{q}_k z^{k-1} + 2\omega^k \times (\dot{q}_k z^{k-1})] \quad (6\text{-}8\text{-}5)$$

This completes the forward Newton-Euler equations. To evaluate the velocities and accelerations of the links, we start at the base. Assuming that the base is not rotating, the *initial conditions* for the forward Newton-Euler recursion are:

$$\omega^0 = 0 \quad (6\text{-}8\text{-}6a)$$

$$\dot{\omega}^0 = 0 \quad (6\text{-}8\text{-}6b)$$

$$v^0 = 0 \quad (6\text{-}8\text{-}6c)$$

$$\dot{v}^0 = -g \quad (6\text{-}8\text{-}6d)$$

Note that we have included the effects of gravity by assuming that the robot base is under a linear acceleration of $-g$. The loading on each link due to gravity is then automatically propagated through the arm by the forward recursive equations.

6-8-2 Backward Newton-Euler Equations

Once the velocities and accelerations of each link are known, the forces and moments acting on each link can then be computed recursively by starting at the tool and working backward to the base, as illustrated in Fig. 6-9.

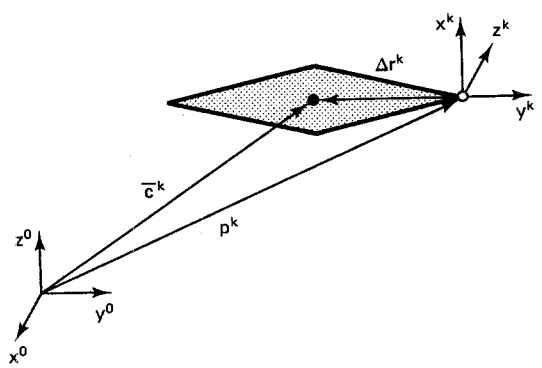


Figure 6-9 Backward Newton-Euler equations.

Let f^k denote the *force* on link k at the origin of L_k due to link $k - 1$. To formulate a concise expression for the link force, it is helpful to first introduce the following variable shown in Fig. 6-9:

$$\Delta r^k \triangleq \bar{c}^k - p^k \quad (6\text{-}8\text{-}7)$$

Recall from Eq. (6-2-12) that $\bar{c}^k$ is the location of the center of mass of link k, while p^k is the location of the origin of the frame, L_k. The force on link k can be expressed recursively in terms of Δr^k by writing a balance of forces equation as follows (Fu et al., 1987):

$$f^k = f^{k+1} + m_k[\dot{v}^k + \dot{\omega}^k \times \Delta r^k + \omega^k \times (\omega^k \times \Delta r^k)] \tag{6-8-8}$$

The term within the square brackets in Eq. (6-8-8) represents the linear acceleration of the center of mass of link k. Next, let n^k denote the *moment* on link k at the origin of L_k due to link $k - 1$. The moment on link k can be expressed in terms of the moment on link $k + 1$ and the forces on link k and link $k + 1$ by writing a balance of moments equation as follows (Fu et al., 1987):

$$n^k = n^{k+1} + (\Delta s^k + \Delta r^k) \times f^k - \Delta r^k \times f^{k+1} + D_k\dot{\omega}^k + \omega^k \times (D_k\omega^k) \tag{6-8-9}$$

Recall that D_k is the inertia tensor of link k. Once the forces and moments acting on each link are known, the applied joint torque vector τ can then be computed. If joint k is revolute, then moment n^k will cause a torque about axis z^{k-1}, whereas if joint k is prismatic, force f^k will cause a force along axis z^{k-1}. In addition, recall that $b_k(\dot{q}_k)$ represents the frictional forces opposing the motion of joint k. These effects can now be combined to produce the following equation for the applied torque at joint k:

$$\tau_k = \xi_k(n^k)^T z^{k-1} + (1 - \xi_k)(f^k)^T z^{k-1} + b_k(\dot{q}_k) \tag{6-8-10}$$

This completes the backward Newton-Euler equations. To evaluate the forces and moments acting on the links, we start at the tool. Let $F^{\text{tool}} = (f^{\text{tool}}, n^{\text{tool}})$ denote the external end-of-arm force and moment vector acting on the tool tip due to a load or contact with the environment. If we regard the environment as link $n + 1$, then the *initial conditions* for the backward Newton-Euler recursion are:

$$f^{n+1} = -f^{\text{tool}} \tag{6-8-11a}$$

$$n^{n+1} = -n^{\text{tool}} \tag{6-8-11b}$$

Thus the effects of an external force and moment acting at the tool tip are included as starting conditions for the backward recursion. If the robot is moving freely in its workspace and is not carrying a load, then $F^{\text{tool}} = 0$. The following algorithm summarizes the recursive Newton-Euler formulation of a dynamic model of an n-axis robot.

Algorithm 6-8-1: Recursive Newton-Euler Equations

1. Set $T_0^0 = I$, $f^{n+1} = -f^{\text{tool}}$, $n^{n+1} = -n^{\text{tool}}$, $v^0 = 0$, $\dot{v}^0 = -g$, $\omega^0 = 0$, $\dot{\omega}^0 = 0$, and $k = 1$.
2. Compute

$$z^{k-1} = R_0^{k-1}\hat{i}^3$$

$$\omega^k = \omega^{k-1} + \xi_k\dot{q}_k z^{k-1}$$

$$\dot{\omega}^k = \dot{\omega}^{k-1} + \xi_k[\ddot{q}_k z^{k-1} + \omega^{k-1} \times (\dot{q}_k z^{k-1})]$$

$$T_0^k = T_0^{k-1} T_{k-1}^k$$

$$\Delta s^k = H_1(T_0^k - T_0^{k-1})i^4$$

$$\dot{v}^k = \dot{v}^{k-1} + \dot{\omega}^k \times \Delta s^k + \omega^k \times (\omega^k \times \Delta s^k)$$
$$+(1 - \xi_k)[\ddot{q}_k z^{k-1} + 2\omega^k \times (\dot{q}_k z^{k-1})]$$

3. Set $k = k + 1$. If $k \leq n$, go to step 2; else, set $k = n$ and continue.

4. Compute

$$\Delta r^k = H_1 T_0^k(\Delta c^k - i^4)$$

$$f^k = f^{k+1} + m_k[\dot{v}^k + \dot{\omega}^k \times \Delta r^k + \omega^k \times (\omega^k \times \Delta r^k)]$$

$$D_k = R_0^k \bar{D}_k (R_0^k)^T$$

$$n^k = n^{k+1} + (\Delta s^k + \Delta r^k) \times f^k - \Delta r^k \times f^{k+1} + D_k \dot{\omega}^k + \omega^k \times (D_k \omega^k)$$

$$\tau_k = \xi_k (n^k)^T z^{k-1} + (1 - \xi_k)(f^k)^T z^{k-1} + b_k(\dot{q}_k)$$

5. Set $k = k - 1$. If $k \geq 1$, go to step 4.

In computing the displacement vectors Δs^k and Δr^k, the 3×4 homogeneous coordinate conversion matrix defined in Eq. (2-4-1) is used. It is clear from Algorithm 6-8-1 that the number of computations required to evaluate the applied torque vector τ is proportional to n, the number of axes. Thus the recursive Newton-Euler formulation has a computational complexity of order $O(n)$. This is in marked contrast to the Lagrange-Euler equations which have a computational complexity of order $O(n^4)$.

The variables in the implementation of the Newton-Euler equations in Algorithm 6-8-1 are all expressed in base coordinates. They can instead be reformulated in terms of link coordinates (Luh et al., 1980). This results in a very efficient algorithm which for $n = 6$ axes requires only 852 multiplications and 738 additions (Hollerbach, 1980). The recursive Newton-Euler formulation, as well as other recursive formulations that have been proposed (Hollerbach, 1980), are not as amenable to simple physical interpretation as the closed-form Lagrange-Euler formulation in Eq. (6-7-1). However, the recursive solutions are much more efficient in terms of computational effort, particularly as the number of axes n increases, and they are ideally suited for computer implementation.

Computers can also be used to derive both the closed-form Lagrange-Euler equations and the recursive Newton-Euler equations *symbolically*. For example, a program called ARM (Algebraic Robot Modeler) has been developed which automates the generation of algorithms for robot dynamics starting from a basic physical description of the manipulator (Neuman and Murray, 1985). By exploiting the kinematic and dynamic structure and removing repetitive calculations, these symbolic modeling techniques have successfully generated highly efficient customized manipulator dynamics algorithms for a variety of robots (Murray and Neuman, 1988).

6-9 DYNAMIC MODEL OF A ONE-AXIS ROBOT (INVERTED PENDULUM)

As a final example of a dynamic model of a robotic arm, consider the one-axis robot or inverted pendulum shown in Fig. 6-10. Again, we assume that the link is a thin cylinder or rod of mass m_1. As a consistency check, the dynamic model of this simple robotic arm is derived using both the Lagrange-Euler formulation and the Newton-Euler formulation.

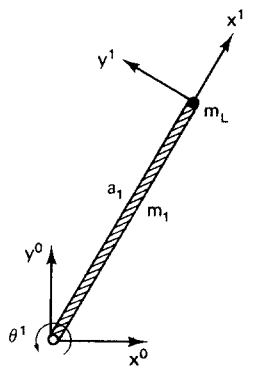


Figure 6-10 A one-axis robot (inverted pendulum).

6-9-1 Lagrange-Euler Formulation

First we develop the dynamic model using the Lagrange-Euler technique of Algorithm 6-4-1. Applying step 1 of Algorithm 6-4-1 yields the link-coordinate diagram shown in Fig. 6-10. Note that link coordinates have been assigned in such a manner that the axes of frame L_1 are aligned with the principal axes of rod m_1. Here the vector of joint variables is $q = \theta_1$. The kinematic parameters for the one-axis robot are listed in Table 6-3.

TABLE 6-3 KINEMATIC PARAMETERS OF THE ONE-AXIS ROBOT (INVERTED PENDULUM)

Joint	d	a	θ	α	Home
1	0	a_1	q_1	0	$\pi/3$

Next we apply steps 3 through 7 of Algorithm 6-4-1 with $i = 1$. Assuming that the link is homogeneous, we see from Fig. 6-10 that the center of mass of link 1 in frame L_1 coordinates is:

$$\Delta c^1 = \left[-\frac{a_1}{2}, 0, 0, 1 \right]^T \tag{6-9-1}$$

Let L_c denote the coordinate frame obtained by translating frame L_1 along axis x^1 by an amount $-a_1/2$. L_c is then at the center of mass of link 1, and the axes of L_c are aligned with the principal axes of link 1. If $\overline{D}_1$ is the inertia tensor of link 1 about its center of mass with respect to frame L_c, then, from Eq. (6-2-2) and Appendix 2:

$$\overline{D}_1 = \left(\frac{m_1 a_1^2}{12}\right) \text{diag}\{0, 1, 1\} \tag{6-9-2}$$

Next, we apply step 5, where $z^0 = i^3$. Using Table 6-3 and Prop. 2-6-1, or using Eq. (6-5-3) directly, the homogeneous coordinate transformation matrix for link 1 is:

$$T_0^1 = \left[\begin{array}{ccc|c} C_1 & -S_1 & 0 & a_1 C_1 \\ S_1 & C_1 & 0 & a_1 S_1 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{6-9-3}$$

Combining Eqs. (6-9-1) and (6-9-3) and recalling Eq. (2-4-1), the center of mass of link 1 in base frame coordinates is:

$$\begin{aligned} \overline{c}^1 &= H_1 T_0^1 \Delta c^1 \\ &= \left[\frac{a_1 C_1}{2}, \frac{a_1 S_1}{2}, 0\right]^T \end{aligned} \tag{6-9-4}$$

Finally, using Eq. (6-9-2) and the rotation part of T_0^1 in Eq. (6-9-3), the inertia tensor of link 1 in base coordinates is:

$$\begin{aligned} D_1 &= R_0^1 \overline{D}_1 (R_0^1)^T \\ &= \left(\frac{m_1 a_1^2}{12}\right) \begin{bmatrix} S_1^2 & -S_1 C_1 & 0 \\ -S_1 C_1 & C_1^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{6-9-5}$$

The Jacobian matrix for link 1 is now computed in step 6. Using Eqs. (6-2-10) and (6-9-4), and recalling that $z^0 = i^3$ and $\xi_1 = 1$, we have the following 6×1 link Jacobian matrix:

$$J^1 = \left[-\frac{a_1 S_1}{2}, \frac{a_1 C_1}{2}, 0 \;\middle|\; 0, 0, 1\right]^T \tag{6-9-6}$$

Partitioning J^1 into A^1 and B^1 as in Eq. (6-2-10), the 1×1 manipulator inertia tensor of the one-axis robot is:

$$\begin{aligned} D(q) &= (A^1)^T m_1 A^1 + (B^1)^T D_1 B^1 \\ &= \frac{m_1 a_1^2}{4} + \frac{m_1 a_1^2}{12} \\ &= \frac{m_1 a_1^2}{3} \end{aligned} \tag{6-9-7}$$

Since $n = 1$, this completes the first pass of Algorithm 6-4-1. Next, we apply steps 9 and 10. Setting $i = 1$ in Eq. (6-4-6) and using Eq. (6-9-7), the velocity coupling matrix for joint 1 is:

$$C^1 = 0 \tag{6-9-8}$$

This is as it should be, since there can clearly be no interaxis velocity coupling due to centrifugal and Coriolis forces in a single-axis robot (see Exercise 6-4-1). If we assume that the y^0 axis in Fig. 6-10 points straight up, directly opposing gravity, then $g = [0, -g_0, 0]^T$, where $g_0 = 9.8062$ m/sec^2. Applying Eq. (6-4-7) and using Eq. (6-9-6), the gravitational loading on joint 1 is:

$$\begin{aligned} h_1 &= g_0 m_1 A_{21}^1 \\ &= \frac{g_0 m_1 a_1 C_1}{2} \end{aligned} \tag{6-9-9}$$

Finally, using Eqs. (6-9-7), (6-9-8), and (6-9-9), we can formulate the dynamic model for the one-axis robotic arm using step 10, as follows:

$$\tau_1 = \left(\frac{m_1 a_1^2}{3}\right)\ddot{q}_1 + \frac{g_0 m_1 a_1 C_1}{2} + b_1(\dot{q}_1) \tag{6-9-10}$$

This is the classical equation of motion of a pendulum, assuming that the pendulum is a thin rod of mass m_1 and length a_1 operating under the influence of friction.

6-9-2 Newton-Euler Formulation

Next we develop the dynamic model of the one-axis robot using the recursive Newton-Euler formulation in Algorithm 6-8-1. To make the analysis compatible with the Lagrange-Euler method, we assume that the robot is moving freely in its workspace and is carrying no payload: $F^{\text{tool}} = 0$. We begin with the forward equations in steps 2 and 3 of Algorithm 6-8-1. Clearly, $z^0 = i^3$ and $\xi_1 = 1$. Thus, from step 2:

$$\omega^1 = \dot{q}_1 i^3 \tag{6-9-11}$$

$$\dot{\omega}^1 = \ddot{q}_1 i^3 \tag{6-9-12}$$

The homogeneous coordinate transformation matrix $T_0^1(q)$ has been previously computed in Eq. (6-9-3). Before we compute the displacement vector Δs^1, it is helpful to first introduce the following intermediate variables as a notational convenience:

$$u^1 \triangleq [C_1, S_1, 0]^T \tag{6-9-13}$$

$$u^2 \triangleq [-S_1, C_1, 0]^T \tag{6-9-14}$$

Note that u^2 is simply the derivative of u^1 with respect to q_1. These vectors arise repeatedly in the subsequent analysis, as do their cross products with i^3:

$$i^3 \times u^1 = u^2 \tag{6-9-15}$$

$$i^3 \times u^2 = -u^1 \tag{6-9-16}$$

Applying step 2 of Algorithm 6-8-1 using Eqs. (6-9-3) and (2-4-1), we have the following displacement of frame L_1 relative to frame L_0:

$$\begin{aligned} \Delta s^1 &= H_1(T_0^1 - T_0^0)i^4 \\ &= a_1 u^1 \end{aligned} \tag{6-9-17}$$

The last equation in step 2 computes the linear acceleration of link 1. Since y^0 in Fig. 6-10 is pointing straight up, we have $g = [0, -g_0, 0]^T$. Now $\xi_1 = 1$, since joint 1 is revolute. Thus, using Eqs. (6-9-11) through (6-9-17) yields:

$$\begin{aligned} \dot{v}^1 &= -g + \dot{\omega}^1 \times \Delta s^1 + \omega^1 \times (\omega^1 \times \Delta s^1) \\ &= -g + a_1\ddot{q}_1(i^3 \times u^1) + \omega^1 \times [a_1\dot{q}_1(i^3 \times u^1)] \\ &= -g + a_1\ddot{q}_1 u^2 + a_1\dot{q}_1(\omega^1 \times u^2) \\ &= -g + a_1\ddot{q}_1 u^2 + a_1\dot{q}_1^2(i^3 \times u^2) \\ &= -g + a_1\ddot{q}_1 u^2 - a_1\dot{q}_1^2 u^1 \end{aligned} \tag{6-9-18}$$

Since $n = 1$, this completes the forward equations.

Next, we examine the Newton-Euler backward equations in steps 3 and 4 of Algorithm 6-8-1. From Eqs. (6-9-1), (6-9-4), and (2-4-1), the displacement between the center of mass of link 1 and frame L_1 is:

$$\begin{aligned} \Delta r^1 &= H_1 T_0^1(\Delta c^1 - i^4) \\ &= \frac{-a_1 u^1}{2} \end{aligned} \tag{6-9-19}$$

We now compute the force acting on link 1, assuming $f^{\text{tool}} = 0$. Applying step 4 and using Eqs. (6-9-11), (6-9-12), (6-9-18), and (6-9-19) yields:

$$\begin{aligned} f^1 &= m_1[\dot{v}^1 + \dot{\omega}^1 \times \Delta r^1 + \omega^1 \times (\omega^1 \times \Delta r^1)] \\ &= m_1\left\{\dot{v}^1 - \left(\frac{a_1\ddot{q}_1}{2}\right)(i^3 \times u^1) + \omega^1 \times \left[\left(-\frac{a_1\dot{q}_1}{2}\right)(i^3 \times u^1)\right]\right\} \\ &= m_1\left[\dot{v}^1 - \frac{a_1\ddot{q}_1 u^2}{2} - \left(\frac{a_1\dot{q}_1}{2}\right)(\omega^1 \times u^2)\right] \\ &= m_1\left[\dot{v}^1 - \frac{a_1\ddot{q}_1 u^2}{2} - \left(\frac{a_1\dot{q}_1^2}{2}\right)(i^3 \times u^2)\right] \\ &= m_1\left(\dot{v}^1 - \frac{a_1\ddot{q}_1 u^2}{2} + \frac{a_1\dot{q}_1^2 u^1}{2}\right) \\ &= m_1\left(-g + a_1\ddot{q}_1 u^2 - a_1\dot{q}_1^2 u^1 - \frac{a_1\ddot{q}_1 u^2}{2} + \frac{a_1\dot{q}_1^2 u^1}{2}\right) \\ &= m_1\left(-g + \frac{a_1\ddot{q}_1 u^2}{2} - \frac{a_1\dot{q}_1^2 u^1}{2}\right) \end{aligned}$$

$$= \frac{m_1 a_1 (\ddot{q}_1 u^2 - \dot{q}_1^2 u^1)}{2} - m_1 g$$

$$= m_1 \left[-\frac{a_1(S_1\ddot{q}_1 + C_1\dot{q}_1^2)}{2}, g_0 + \frac{a_1(C_1\ddot{q}_1 - S_1\dot{q}_1^2)}{2}, 0 \right]^T \qquad (6\text{-}9\text{-}20)$$

The inertia tensor for link 1 was previously computed in Eq. (6-9-5).

Next, we compute the moment acting on link 1, assuming $n^{\text{tool}} = 0$. Applying step 4 and using Eqs. (6-9-11), (6-9-12), (6-9-17), (6-9-19), and (6-9-20) yields:

$$\begin{aligned}
n^1 &= (\Delta s^1 + \Delta r^1) \times f^1 + D_1 \dot{\omega}^1 + \omega^1 \times (D_1 \omega^1) \\
&= \left(a_1 u^1 - \frac{a_1 u^1}{2}\right) \times f^1 + \frac{m_1 a_1^2 \ddot{q}_1 i^3}{12} + \frac{m_1 a_1^2 \dot{q}_1^2 (i^3 \times i^3)}{12} \\
&= \left(\frac{a_1}{2}\right)(u^1 \times f^1) + \frac{m_1 a_1^2 \ddot{q}_1 i^3}{12} \\
&= \left(\frac{a_1}{2}\right)(C_1 f_2^1 - S_1 f_1^1) i^3 + \frac{m_1 a_1^2 \ddot{q}_1 i^3}{12} \\
&= \left(\frac{m_1 a_1}{2}\right)[C_1(g_0 + a_1(C_1\ddot{q}_1 - S_1\dot{q}_1^2) + S_1 a_1(S_1\ddot{q}_1 + C_1\dot{q}_1^2)]i^3 + \frac{m_1 a_1^2 \ddot{q}_1 i^3}{12} \\
&= \left[\frac{g_0 m_1 a_1 C_1}{2} + \frac{m_1 a_1^2 \ddot{q}_1}{4}\right] i^3 + \frac{m_1 a_1^2 \ddot{q}_1 i^3}{12} \\
&= \left[\left(\frac{m_1}{3}\right) a_1^2 \ddot{q}_1 + \frac{g_0 m_1 a_1 C_1}{2}\right] i^3 \qquad (6\text{-}9\text{-}21)
\end{aligned}$$

Finally, the applied torque for joint 1 using step 4 is:

$$\tau_1 = \left(\frac{m_1 a_1^2}{3}\right)\ddot{q}_1 + \frac{g_0 m_1 a_1 C_1}{2} + b_1(\dot{q}_1) \qquad (6\text{-}9\text{-}22)$$

Since $n = 1$, this completes Algorithm 6-8-1. Comparing Eq. (6-9-22) with Eq. (6-9-10), we see that the recursive Newton-Euler method generates the same dynamic model as the Lagrange-Euler method. For this simple example where $n = 1$, the Newton-Euler derivation "appears" to involve as much work as the Lagrange-Euler derivation. However, as the number of axes n increases, the computational work required to evaluate the Lagrange-Euler equations grows much more rapidly than the corresponding work for the Newton-Euler technique. Furthermore, the recursive Newton-Euler equations in Algorithm 6-8-1 are ideally suited for implementation on a computer. It is important to note that it is only the *value* of τ that needs to be computed, there is no need to derive a closed-form *expression* for τ as was done here to facilitate a comparison of the two methods.

6-10 PROBLEMS

6-1. Consider the cone-shaped link of mass m shown in Fig. 6-11. Let L_c be the frame obtained by translating frame $L_0 = \{x^0, y^0, z^0\}$ to the center of mass of the cone. Find the inertia tensor $\overline{D}$ of this link about its center of mass expressed with respect to frame L_c.

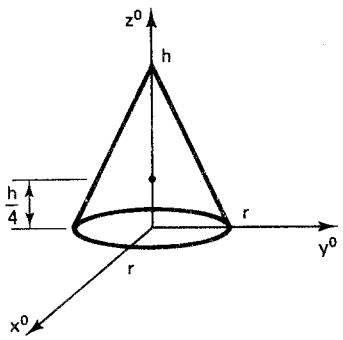


Figure 6-11 A cone-shaped link.

6-2. Consider the two-axis polar-coordinate robot shown in Fig. 6-12, where links 1 and 2 are thin cylinders or rods of mass m_1 and m_2, respectively. The vector of joint variables is $q = [\theta_1, d_2]^T$. When joint 1 is activated, the entire robot, including rod m_1, rotates. When joint 2 is activated, the entire rod of length l_2 slides radially in and out.

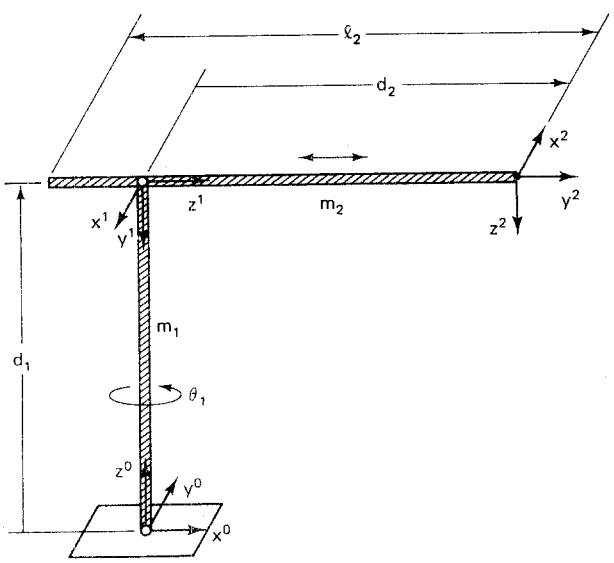


Figure 6-12 A two-axis polar-coordinate robot.

The kinematic parameters for the two-axis polar-coordinate robot are listed in Table 6-4. Find the homogeneous coordinate transformation matrices $T_0^1(q)$ and $T_0^2(q)$ of this robot.

TABLE 6-4 KINEMATIC PARAMETERS OF THE TWO-AXIS POLAR COORDINATE ROBOT

Joint	d	a	θ	α	Home
1	d_1	0	q_1	$-\pi/2$	$-\pi/2$
2	q_2	0	π	$\pi/2$	$l_3/2$

6-3. Assuming that the links of the polar-coordinate robot in Fig. 6-12 are thin homogeneous cylinders, find the center of mass $\bar{c}^k(q)$ with respect to the base frame L_0 for $k = 1, 2$.

6-4. Find the total potential energy $U(q)$ of the polar-coordinate robot in Fig. 6-12, assuming $g = [0, 0, -g_0]^T$.

6-5. For the polar-coordinate robot in Fig. 6-12, find the link Jacobian matrix $J^k(q)$ for $k = 1, 2$.

6-6. For the polar-coordinate robot in Fig. 6-12, find $\bar{D}_k$, the inertia tensor of link k about its center of mass expressed with respect to frame L_k for $k = 1, 2$. Here link k is assumed to be a thin cylinder of mass m_k for $k = 1, 2$.

6-7. For the polar-coordinate robot in Fig. 6-12, find $D_k(q)$, the inertia tensor of link k about its center of mass expressed with respect to frame L_0 for $k = 1, 2$.

6-8. Find the manipulator inertia tensor $D(q)$ of the polar-coordinate robot in Fig. 6-12.

6-9. Find the total kinetic energy $T(q, \dot{q})$ of the polar-coordinate robot in Fig. 6-12.

6-10. Find the Lagrangian function $L(q, \dot{q})$ of the polar-coordinate robot in Fig. 6-12.

6-11. For the polar-coordinate robot in Fig. 6-12, find the velocity coupling matrix $C^k(q)$ for $k = 1, 2$.

6-12. Use the results of Prob. 6-11 to identify the centrifugal forces acting on joint k for $k = 1, 2$.

6-13. Use the results of Prob. 6-11 to identify the Coriolis forces acting on joint k for $k = 1, 2$.

6-14. For the polar-coordinate robot in Fig. 6-12, find the gravity loading $h_k(q)$ for $k = 1, 2$, assuming $g = [0, 0, -g_0]^T$.

6-15. Find the generalized forces acting on joints 1 and 2 of the polar-coordinate robot in Fig. 6-12.

6-16. Find the complete dynamic model of the polar-coordinate robot in Fig. 6-12.

REFERENCES

ASADA, H., and J. E. SLOTIN (1986). *Robot Analysis and Control,* Wiley: New York.

BRADY, M., J. M. HOLLERBACH, T. L. JOHNSON, T. LOZANO-PEREZ, and M. T. MASON (1982). *Robot Motion: Planning and Control,* MIT Press: Cambridge, Mass.

CRAIG, J. (1986). *Introduction to Robotics: Mechanics and Control,* Addison-Wesley: Reading, Mass.

FU, K. S., R. C. GONZALEZ, and C. S. G. LEE (1987). *Robotics: Control, Sensing, Vision, and Intelligence,* McGraw-Hill: New York.

GEAR, C. W. (1971). *Numerical Initial Value Problems in Ordinary Differential Equations,* Prentice Hall: Englewood Cliffs, N.J.

HOLLERBACH, J. M. (1980). "A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity," *IEEE Trans. Syst., Man, Cybern.*, Vol. SMC-10, pp. 730–736.

KLAFTER, R. D., T. A. CHMIELEWSKI, and M. NEGIN (1989). *Robotic Engineering: An Integrated Approach*, Prentice Hall: Englewood Cliffs, N.J.

LUH, J. Y. S., M. W. WALKER, and R. P. C. PAUL (1980). "On-line computational scheme for mechanical manipulators," *J. Dynamic Systems, Measurement and Control*, Vol. 102, pp. 69–76.

MURRAY, J. J., and C. P. NEUMAN (1988). "Organizing customized robot dynamics algorithms for efficient numerical evaluation," *IEEE Trans. Syst., Man, and Cybern.*, Vol. SMC-18, pp. 115–125.

NEUMAN, C. P., and J. J. MURRAY (1985). "Computational robot dynamics: Foundations and applications," *J. Robotic Syst.*, Vol.2, pp. 425–452.

PAUL, R. P. (1981). *Robot Manipulators: Mathematics, Programming, and Control*, MIT Press: Cambridge, Mass.

TORBY, B. J. (1984). *Advanced Dynamics for Engineers*, Holt, Rinehart & Winston: New York.

WOLOVICH, W. A. (1987). *Robotics: Basic Analysis and Design*, Holt, Rinehart & Winston: New York.

7

Robot Control

In Chap. 4 we examined the problem of planning a robot trajectory in tool-configuration space. Once a nominal trajectory is determined, there remains the problem of issuing the commands to the joint actuators that will cause the manipulator to faithfully "track" or follow the planned trajectory. This is called the *robot control* problem, and numerous techniques have been proposed for its solution. One approach is to convert the tool-configuration trajectory $w(t)$ to a corresponding joint-space trajectory $q(t)$, through the inverse kinematics equations, and then differentiate to determine the rates $\dot{q}(t)$ at which the individual joints should be driven as in Chap. 5. A variation of this technique is the resolved-motion rate-control method, which uses a generalized inverse of the tool-configuration Jacobian matrix. Both of these approaches implicitly assume that hardware is available to regulate the *speed* of each joint.

An alternative approach to controlling a robotic arm is to instead regulate the joint *torque* τ. Torque-based control techniques build directly on the dynamic models developed in Chap. 6 and are the focus of this chapter. The challenges inherent in the robot control problem are first outlined. A state-space formulation of the dynamic model of a robotic manipulator is then developed. The simplest solutions to the robot state equation, the constant solutions or points of equilibrium, are then investigated using Liapunov stability techniques. A brief review of linear feedback systems is then presented and these techniques are used to analyze single-axis PID control, a design method that is widely used in current commercial robots. More sophisticated nonlinear methods are then examined including PD control with gravity compensation, computed-torque control, variable-structure control, and impedance control.

7-1 THE CONTROL PROBLEM

High-performance control of robotic manipulators through torque regulation is a difficult task. The difficulty arises from the complexity of the dynamic model of the arm. Recall that the equations of motion of a robotic arm are of the following general form:

$$D(q)\ddot{q} + c(q, \dot{q}) + h(q) + b(\dot{q}) = \tau \tag{7-1-1}$$

For an n-axis robot, the manipulator is modeled as a simultaneous system of n highly coupled nonlinear second-order differential equations. Several features of current commercial robots simplify the control task somewhat. The first is the observation that in many applications the robot does not have to be driven at high speed. Since $\dot{q}$ represents the joint velocities, the following is a rough measure of the overall speed of the manipulator:

$$\text{Arm speed} = \|\dot{q}\| \tag{7-1-2}$$

One of the most complex terms in the dynamic model is the velocity coupling term $c(q, \dot{q})$, which is generated by centrifugal and Coriolis forces. Recall from Eq. (6-4-8) that the kth component of the velocity coupling vector is of the form $c_k(q, \dot{q}) = \dot{q}^T C^k(q)\dot{q}$. Consequently, for each point q in joint space:

$$c(q, \dot{q}) \to 0 \qquad \text{as} \qquad \|\dot{q}\| \to 0 \tag{7-1-3}$$

Therefore, if the speed of the robotic arm is reduced sufficiently, the dynamic model simplifies in the sense that the effects of centrifugal and Coriolis force become small in comparison with the terms that do not depend on $\dot{q}$. If the robot is operated in a single-axis mode where the joints are activated one at a time, rather than simultaneously, then velocity coupling due to Coriolis force (but not centrifugal force) disappears completely, independent of the arm speed.

The friction term $b(\dot{q})$ also depends on joint speed, and from Eq. (6-3-3), we see that $b(0) = 0$ since sgn $(0) = 0$. However, the torque due to friction cannot be made arbitrarily small by operating the arm at reduced speed. This is because $b(\dot{q})$ has a jump discontinuity at $\dot{q} = 0$. Indeed, it is clear from Fig. 6-3 that the *relative* effects of static friction become more pronounced as the arm speed is reduced.

Another characteristic of many commercial robots that makes them somewhat easier to control is the use of gear reduction between the actuators and the links (Asada and Slotine, 1986). Let M_k denote the *gear reduction ratio* from the actuator shaft to the load shaft of joint k. Then the variable component of the arm inertia, referred to the actuator for joint k, is reduced by a factor of M_k^2. Similarly, the torque due to velocity coupling, $c_k(q, \dot{q})$, and the loading torque due to gravity, $h_k(q)$, are reduced by a factor of M_k when referred to the joint actuator. Gear reduction ratios of commercial robots can sometimes be quite substantial. For example, the vector of gear reduction ratios for the five-axis Rhino XR-3 robot is (Hendrickson and Sandhu, 1985):

$$M = [264.4, 528.8, 528.8, 528.8, 384.0]^T \tag{7-1-4}$$

Thus the change in inertia, as seen by the shoulder motor, due to motion of the elbow, wrist pitch, or wrist roll joints is reduced by a factor of more than 10^5. Clearly, this educational robot is designed to be quite "stiff" in comparison with a highly dynamical control system. This makes the robot easier to control, but it reduces its operating speed and, potentially, its productivity.

Most current commercial robots have relatively simple controllers, use gear reduction, and operate at modest speeds. This reflects the fact that the technology is still far from mature. Current trends in robot design are in the direction of *direct-drive* robots, such as the Adept One SCARA robot, where special high-torque actuators drive the joints directly (An et al., 1988). The elimination of gears and other power transmission mechanisms reduces friction and backlash and also allows the robot joints to operate at considerably higher speeds. Improved dynamic performance of manipulators is clearly needed if industrial robots are to live up to their potential for increasing productivity and improving production quality. For this reason, sophisticated control of robotic manipulators using realistic dynamic models, although a challenging task, is one that must be addressed.

Development of control algorithms for robot manipulators is currently an area of active research. This chapter does not include a complete survey of current robot control methods, because many of the algorithms proposed in the literature are very sophisticated, still under active development and rely on techniques beyond the scope of this book. Instead, a *sampling* of robot control algorithms, with a bias toward the simpler techniques, is presented. Given the nature of the topic, the discussion in this chapter assumes that the reader has familiarity with elementary differential equations and the basics of linear control systems, including the Laplace transform and transfer functions (Ogata, 1970; Kuo, 1982).

7-2 STATE EQUATIONS

The dynamic model of a robotic arm is formulated in Eq. (7-1-1) as a second-order system of n differential equations. To facilitate control of the arm, it is helpful to reformulate the equations of motion as a first-order system of $2n$ equations called *state equations*. The key to transforming to the state-space form is the isolation of the acceleration vector $\ddot{q}$ in Eq. (7-1-1). This can be easily done because the manipulator inertia tensor $D(q)$ is positive-definite and therefore nonsingular. The following state-space model of a robotic arm includes both a dynamic state equation and a kinematic output equation.

Proposition 7-2-1: State-Space Representation. Define $x^T \triangleq [q^T, v^T]$, where $v \triangleq \dot{q}$. Then the equation of motion of a robotic arm in Eq. (7-1-1) can be represented by the following first-order state-space model:

$$\dot{q} = v$$

$$\dot{v} = D^{-1}(q)[\tau - h(q) - c(q, v) - b(v)]$$

$$y = w(q)$$

Proof. By definition, $\dot{q} = v$. Using Eq. (7-1-1) and the definition of v, we

then have:

$$\begin{aligned}\dot{v} &= \ddot{q} \\ &= D^{-1}(q)[\tau - c(q, \dot{q}) - h(q) - b(\dot{q})] \\ &= D^{-1}(q)[\tau - c(q, v) - h(q) - b(v)]\end{aligned}$$

If $w(q)$ represents the tool-configuration function in Def. 3-3-1, then the robot output y is:

$$y = w(q)$$

We refer to the vector $x(t) \in \mathbf{R}^{2n}$ as the *state* of the robotic arm at time t. The output equation $y = w(q)$, shows that the robot tool configuration depends on only the first n components of the state vector x. Recall from Chap. 3 that $w(q)$ is a vector in $\mathbf{R}^6$ representing the tool-tip position and tool orientation. We can summarize Prop. 7-2-1 by saying that a robotic arm with n axes can be modeled by a first-order nonlinear system of dimension $2n$. This system has n inputs which consist of the torques or forces delivered by the actuators, and six outputs which specify the configuration of the tool. A block diagram of the state-space model of a robotic arm is shown in Fig. 7-1.

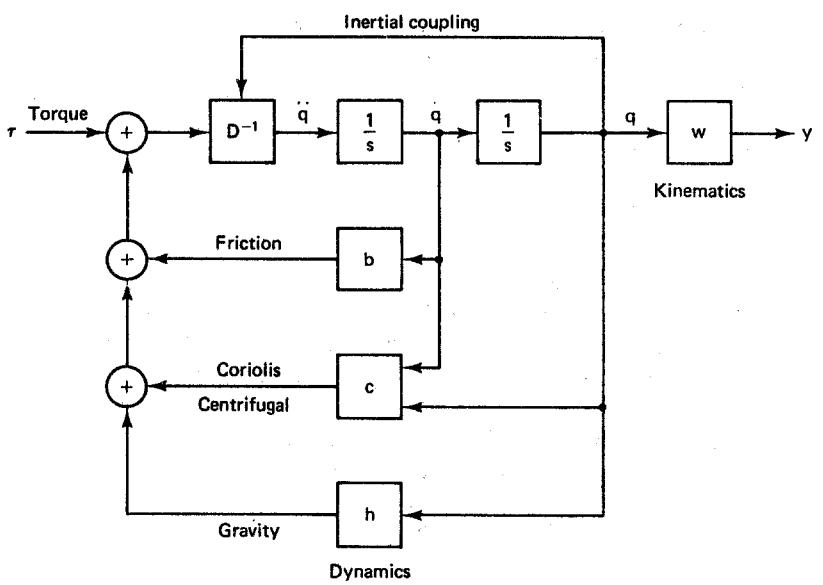


Figure 7-1 State-space model of a robotic arm.

7-2-1 A One-Axis Robot (Inverted Pendulum)

As an example of a state-space model, consider the one-axis robot or inverted pendulum shown in Fig. 7-2. Here the single link is assumed to be a thin homogeneous rod of mass m_1 and length a_1. We also assume that a "load" carried by the one-axis robot can be modeled as a point mass m_L at the end of the arm.

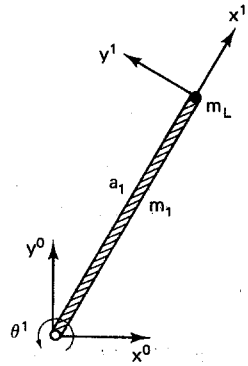


Figure 7-2 A one-axis robot (inverted pendulum).

The dynamic model of this one-axis robot was developed in Sec. 6-9 for the case $m_L = 0$. The inclusion of the load mass increases the moment of inertia by $m_L a_1^2$ and the gravity loading by $g_0 m_L a_1 C_1$. Adding these effects to Eq. (6-9-10), the equation of motion of the arm with a load is:

$$\tau_1 = \left(\frac{m_1}{3} + m_L\right) a_1^2 \ddot{q}_1 + g_0 \left(\frac{m_1}{2} + m_L\right) a_1 C_1 + b_1(\dot{q}_1) \qquad (7\text{-}2\text{-}1)$$

There are no velocity coupling terms due to Coriolis and centrifugal forces because there is only one axis. If we define the state vector as $x^T = [q_1, v]$ where $v = \dot{q}_1$, then from Prop. 7-2-1 the state equation of this system is:

$$\dot{q} = v \qquad (7\text{-}2\text{-}2a)$$

$$\dot{v} = \frac{\tau_1 - g_0(m_1/2 + m_L)a_1 C_1 - b_1(v)}{(m_1/3 + m_L)a_1^2} \qquad (7\text{-}2\text{-}2b)$$

Note that the manipulator inertia tensor in this case is a constant positive 1×1 matrix whose inverse is $D^{-1}(q) = 1/[(m_1/3 + m_L)a_1^2]$. The output equation in Prop. 7-2-1 specifies the position and orientation of the tool through the tool-configuration function $w(q)$ defined in Def. 3-3-1. From Fig. 7-2 we see that the tool-tip position is $p(q) = [a_1 C_1, a_1 S_1, 0]^T$ and the approach vector is $r^3(q) = [0, 0, 1]^T$. Since there is no tool roll joint, we scale $r^3(q)$ by $\exp(0/\pi) = 1$. Thus the kinematic output equation of the one-axis robot is:

$$y = [a_1 C_1, a_1 S_1, 0, 0, 0, 1]^T \qquad (7\text{-}2\text{-}3)$$

7-2-2 A Two-Axis Planar Articulated Robot

As a second example of a state space model, consider the two-axis planar articulated robot in Fig. 7-3. Here the kth link is assumed to be a thin homogeneous rod of mass m_k and length a_k.

The dynamic model of this two-axis robot was developed in Sec. 6-5 using the Lagrange-Euler formulation. From Eqs. (6-5-17) and (6-5-20), the equations of mo-

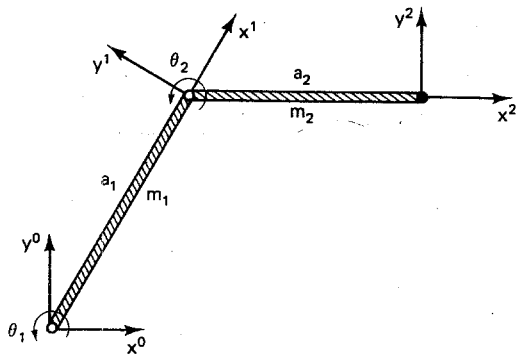


Figure 7-3 A two-axis planar articulated robot.

tion of the arm are:

$$\tau_1 = \left[\left(\frac{m_1}{3} + m_2\right)a_1^2 + m_2 a_1 a_2 C_2 + \frac{m_2 a_2^2}{3}\right]\ddot{q}_1 + \left(\frac{m_2 a_1 a_2 C_2}{2} + \frac{m_2 a_2^2}{3}\right)\ddot{q}_2 - m_2 a_1 a_2 S_2\left(\dot{q}_1\dot{q}_2 + \frac{\dot{q}_2^2}{2}\right) + g_0\left[\left(\frac{m_1}{2} + m_2\right)a_1 C_1 + \frac{m_2 a_2 C_{12}}{2}\right] + b_1(\dot{q}_1) \tag{7-2-4a}$$

$$\tau_2 = \left(\frac{m_2 a_1 a_2 C_2}{2} + \frac{m_2 a_2^2}{3}\right)\ddot{q}_1 + \frac{m_2 a_2^2 \ddot{q}_2}{3} + \frac{m_2 a_1 a_2 S_2 \dot{q}_1^2}{2} + \frac{g_0 m_2 a_2 C_{12}}{2} + b_2(\dot{q}_2) \tag{7-2-4b}$$

If we define the state as $x^T = [q^T, v^T]$ where $v = \dot{q}$, then the state equation can be obtained as in Prop. 7-2-1. Since $n = 2$, the manipulator inertia tensor $D(q)$ is a symmetric 2×2 matrix. Consequently, the inverse of $D(q)$ can be formulated by interchanging the diagonal elements, changing the signs of the off-diagonal elements, and dividing by the determinant. That is:

$$D^{-1}(q) = \frac{1}{\Delta}\begin{bmatrix} D_{22} & -D_{12} \\ -D_{21} & D_{11} \end{bmatrix} \tag{7-2-5}$$

From the coefficients of the joint accelerations in Eq. (7-2-4), or from Eq. (6-5-14) directly, we see that $D(q)$ is symmetric and the distinct components of $D(q)$ are:

$$D_{11}(q) = \left(\frac{m_1}{3} + m_2\right)a_1^2 + m_2 a_1 a_2 C_2 + \frac{m_2 a_2^2}{3} \tag{7-2-6a}$$

$$D_{12}(q) = \frac{m_2 a_1 a_2 C_2}{2} + \frac{m_2 a_2^2}{3} \tag{7-2-6b}$$

$$D_{22}(q) = \frac{m_2 a_2^2}{3} \tag{7-2-6c}$$

Using Eq. (7-2-6), the determinant of the manipulator inertia tensor is:

$$\begin{aligned}\Delta(q) &= D_{11}D_{22} - D_{12}D_{21} \\ &= \left[\left(\frac{m_1}{3} + m_2\right)a_1^2 + m_2a_1a_2C_2 + \frac{m_2a_2^2}{3}\right]\frac{m_2a_2^2}{3} - \left(\frac{m_2a_1a_2C_2}{2} + \frac{m_2a_2^2}{3}\right)^2 \\ &= \left(\frac{m_1}{3} + m_2\right)\frac{a_1^2m_2a_2^2}{3} - \frac{(m_2a_1a_2C_2)^2}{4} \\ &= \frac{m_1m_2(a_1a_2)^2}{9} + \frac{(m_2a_1a_2)^2}{3} - \frac{(m_2a_1a_2C_2)^2}{4} \\ &= m_2(a_1a_2)^2\left[\frac{m_1}{9} + m_2\left(\frac{1}{3} - \frac{C_2^2}{4}\right)\right] \end{aligned} \tag{7-2-7}$$

Note that $\Delta(q) > 0$ for all q, as expected, since $D(q)$ is positive-definite. The state equations of the two-axis planar articulated robot can now be formulated in terms of $\Delta(q)$ and the components of $D(q)$ using Eq. (7-2-5) and Prop. 7-2-1 as follows:

$$\dot{q}_1 = v_1 \tag{7-2-8a}$$

$$\dot{q}_2 = v_2 \tag{7-2-8b}$$

$$\begin{aligned}\dot{v}_1 = \frac{1}{\Delta(q)}\Bigg(&D_{22}(q)\left\{\tau_1 + m_2a_1a_2S_2\left(v_1v_2 + \frac{v_2^2}{2}\right) - g_0\left[\left(\frac{m_1}{2} + m_2\right)a_1C_1 + \frac{m_2a_2C_{12}}{2}\right]\right. \\ &\left. - b_1(v_1)\right\} - D_{12}(q)\left[\tau_2 - \frac{m_2a_1a_2S_2v_1^2}{2} - \frac{g_0m_2a_2C_{12}}{2} - b_2(v_2)\right]\Bigg)\end{aligned} \tag{7-2-8c}$$

$$\begin{aligned}\dot{v}_2 = \frac{1}{\Delta(q)}\Bigg(&-D_{21}(q)\left\{\tau_1 + m_2a_1a_2S_2\left(v_1v_2 + \frac{v_2^2}{2}\right)\right. \\ &\left. - g_0\left[\left(\frac{m_1}{2} + m_2\right)a_1C_1 + \frac{m_2a_2C_{12}}{2}\right] - b_1(v_1)\right\} \\ &+ D_{11}(q)\left[\tau_2 - \frac{m_2a_1a_2S_2v_1^2}{2} - \frac{g_0m_2a_2C_{12}}{2} - b_2(v_2)\right]\Bigg)\end{aligned} \tag{7-2-8d}$$

To determine the output equation of the two-axis planar manipulator, first note from Fig. 7-3, or from Eq. (3-7-2) directly, that the tool-tip position is $p(q) = [a_1C_1 + a_2C_{12},\ a_1S_1 + a_2S_{12},\ 0]^T$. Similarly, the approach vector is $r^3(q) = [0,\ 0,\ 1]^T$ and again there is no tool roll joint. Thus the kinematic output equation of the two-axis planar robot is:

$$y = [a_1C_1 + a_2C_{12},\ a_1S_1 + a_2S_{12},\ 0,\ 0,\ 0,\ 1]^T \tag{7-2-9}$$

7-2-3 A Three-Axis SCARA Robot

As a final example of a state-space model, consider the three-axis SCARA robot in Fig. 7-4. Here the kth link is again assumed to be a thin homogeneous rod of mass m_k and length a_k.

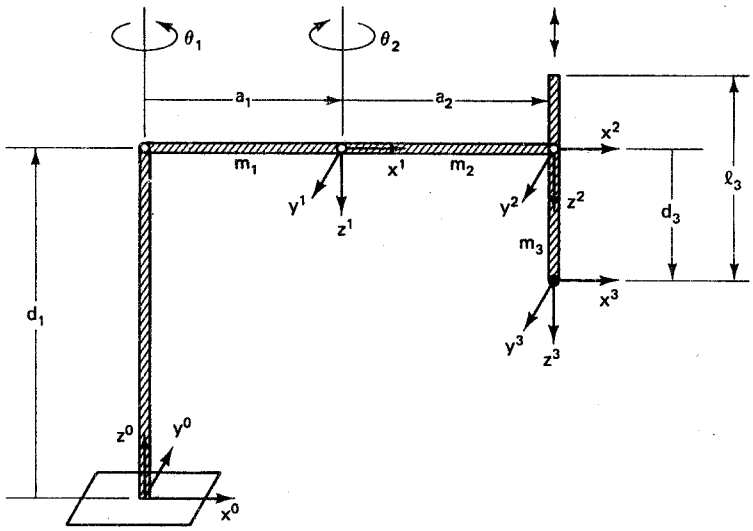


Figure 7-4 A three-axis SCARA robot.

The dynamic model of this three-axis robot was developed in Sec. 6-6 using the Lagrange-Euler formulation. From Eqs. (6-6-25), (6-6-28), and (6-6-31), the equations of motion of the arm are:

$$\tau_1 = \left[\left(\frac{m_1}{3} + m_2 + m_3\right)a_1^2 + (m_2 + 2m_3)a_1a_2C_2 + \left(\frac{m_2}{3} + m_3\right)a_2^2\right]\ddot{q}_1$$
$$- \left[\left(\frac{m_2}{2} + m_3\right)a_1a_2C_2 + \left(\frac{m_2}{3} + m_3\right)a_2^2\right]\ddot{q}_2 + b_1(\dot{q}_1)$$
$$- a_1a_2S_2\left[(m_2 + 2m_3)\dot{q}_1\dot{q}_2 - \left(\frac{m_2}{2} + m_3\right)\dot{q}_2^2\right] \qquad (7\text{-}2\text{-}10a)$$

$$\tau_2 = -\left[\left(\frac{m_2}{2} + m_3\right)a_1a_2C_2 + \left(\frac{m_2}{3} + m_3\right)a_2^2\right]\ddot{q}_1 + \left(\frac{m_2}{3} + m_3\right)a_2^2\ddot{q}_2$$
$$+ \left(\frac{m_2}{2} + m_3\right)a_1a_2S_2\dot{q}_1^2 + b_2(\dot{q}_2) \qquad (7\text{-}2\text{-}10b)$$

$$\tau_3 = m_3\ddot{q}_3 - g_0m_3 + b_3(\dot{q}_3) \qquad (7\text{-}2\text{-}10c)$$

If we define the state as $x^T = [q^T, v^T]$, where $v = \dot{q}$, then the state equation can again be obtained as in Prop. 7-2-1. Since $n = 3$, the manipulator inertia tensor is a 3×3 symmetric matrix. From the coefficients of the joint accelerations in Eq. (7-2-10), or from Eq. (6-6-22) directly, we see that $D(q)$ has the following special *block diagonal* structure:

$$D(q) = \left[\begin{array}{cc:c} D_{11} & D_{12} & 0 \\ D_{21} & D_{22} & 0 \\ \hdashline 0 & 0 & m_3 \end{array}\right] \qquad (7\text{-}2\text{-}11)$$

Here $D(q)$ is symmetric with $D_{21}(q) = D_{12}(q)$, and the remaining nonconstant terms of $D(q)$ are:

$$D_{11}(q) = \left(\frac{m_1}{3} + m_2 + m_3\right)a_1^2 + (m_2 + 2m_3)a_1a_2C_2 + \left(\frac{m_2}{3} + m_3\right)a_2^2 \tag{7-2-12a}$$

$$D_{12}(q) = -\left[\left(\frac{m_2}{2} + m_3\right)a_1a_2C_2 + \left(\frac{m_2}{3} + m_3\right)a_2^2\right] \tag{7-2-12b}$$

$$D_{22}(q) = \left(\frac{m_2}{3} + m_3\right)a_2^2 \tag{7-2-12c}$$

Since $D(q)$ is block diagonal, its inverse can be obtained by inverting the two diagonal blocks separately. The inverse of the 1×1 block in the lower right corner of Eq. (7-2-11) is simply $1/m_3$. To obtain the inverse of the 2×2 block in the upper left corner of Eq. (7-2-11), we use the same formula as in Eq. (7-2-5). However, in this case the determinant of the 2×2 diagonal block is:

$$\begin{aligned}\Delta(q) &= D_{11}D_{22} - D_{12}D_{21} \\ &= \left[\left(\frac{m_1}{3} + m_2 + m_3\right)a_1^2 + (m_2 + 2m_3)a_1a_2C_2 + \left(\frac{m_2}{3} + m_3\right)a_2^2\right]\left(\frac{m_2}{3} + m_3\right)a_2^2 \\ &\quad - \left[\left(\frac{m_2}{2} + m_3\right)a_1a_2C_2 + \left(\frac{m_2}{3} + m_3\right)a_2^2\right]^2 \\ &= \left(\frac{m_2}{3} + m_3\right)^2(a_1a_2)^2 + \frac{(m_1a_1a_2)^2}{3} - \left(\frac{m_2}{2} + m_3\right)^2(a_1a_2C_2)^2 \\ &= (a_1a_2)^2\left[\frac{m_1^2}{3} + \left(\frac{m_2}{3} + m_3\right)^2(1 - C_2^2)\right]\end{aligned} \tag{7-2-13}$$

Again note that $m_3\Delta(q) > 0$ for all q, as expected, since $D(q)$ is positive-definite. The state equations of the three-axis SCARA robot can now be formulated in terms of $\Delta(q)$ and the components of $D(q)$ using Eq. (7-2-5) and Prop. 7-2-1 as follows:

$$\dot{q}_1 = v_1 \tag{7-2-14a}$$

$$\dot{q}_2 = v_2 \tag{7-2-14b}$$

$$\dot{q}_3 = v_3 \tag{7-2-14c}$$

$$\begin{aligned}\dot{v}_1 = \frac{1}{\Delta(q)}\Bigg(&D_{22}(q)\left\{\tau_1 - b_1(v_1) + a_1a_2S_2\left[(m_2 + 2m_3)v_1v_2 - \left(\frac{m_2}{2} + m_3\right)v_2^2\right]\right\} \\ &- D_{12}(q)\left[\tau_2 - \left(\frac{m_2}{2} + m_3\right)a_1a_2S_2v_1^2 - b_2(v_2)\right]\Bigg)\end{aligned} \tag{7-2-14d}$$

$$\dot{v}_2 = \frac{1}{\Delta(q)}\left(-D_{21}(q)\left\{\tau_1 - b_1(v_1) + a_1a_2S_2\left[(m_2 + 2m_3)v_1v_2 - \left(\frac{m_2}{2} + m_3\right)v_2^2\right]\right\}\right.$$

$$+ D_{11}(q)\left[\tau_2 - \left(\frac{m_2}{2} + m_3\right)a_1 a_2 S_2 v_1^2 - b_2(v_2)\right]\Bigg) \tag{7-2-14e}$$

$$\dot{v}_3 = \frac{\tau_3 + g_0 m_3 - b_3(v_3)}{m_3} \tag{7-2-14f}$$

To determine the output equation of the three-axis SCARA robot, note from Fig. 7-4, or from Eq. (3-5-1) directly, that the tool-tip position is $p(q) = [a_1C_1 + a_2C_{12}, a_1S_1 + a_2S_{12}, d_1 - q_3]^T$. In this case the approach vector is $r^3(q) = [0, 0, -1]^T$ and again there is no tool roll joint. Thus the kinematic output equation of the three-axis SCARA robot is:

$$y = [a_1C_1 + a_2C_{12}, a_1S_1 + a_2S_{12}, d_1 - q_3, 0, 0, -1]^T \tag{7-2-15}$$

7-3 CONSTANT SOLUTIONS

To analyze the nonlinear state equations of a robotic arm, we first introduce some fundamental concepts that are applicable to nonlinear systems in general. These techniques allow us to determine certain *qualitative* features of solutions of a nonlinear system without actually obtaining explicit expressions for those solutions. This is an important feature because, for nonlinear systems, explicit closed-form expressions for the solution may not exist. In order to simplify the notation, we restrict our attention to a nonlinear system of the following generic form:

$$\dot{x} = g(x, u) \tag{7-3-1}$$

For convenience, we refer to the system of equations represented by Eq. (7-3-1) as the *nonlinear system S*. Here the independent time variable, denoted t, is left implicit. For each $t \geq 0$, the vector $x(t) \in \mathbf{R}^m$ is the *state* of S at time t, and the vector $u(t) \in \mathbf{R}^n$ is the *input* to S at time t. We assume that for the class of inputs of interest, the system S has a unique solution $x(t)$ for $t \geq 0$ satisfying the *initial condition:*

$$x(0) = x^0 \tag{7-3-2}$$

Here $x^0 \in \mathbf{R}^m$ is called the *initial state*. The system S is a *nonautonomous* system because it has an input $u(t)$ which can vary with time. If we restrict our attention to the case of constant inputs, then S becomes an *autonomous* system. Autonomous systems are easier to analyze because they often possess one or more *constant* solutions called *equilibrium points*.

Definition 7-3-1: Equilibrium Point. Let $u(t) = r$ for $t \geq 0$ for some $r \in \mathbf{R}^n$. Then $\hat{x} \in \mathbf{R}^m$ is an *equilibrium point* of the system S associated with the input $u(t) = r$ if and only if

$$g(\hat{x}, r) = 0$$

It is evident from Eq. (7-3-1) that $\dot{x}(t) = 0$ at each equilibrium point of S.

Thus if $x(0) = \hat{x}$, where $\hat{x}$ is an equilibrium point, then $x(t) = \hat{x}$ for $t \geq 0$. That is, equilibrium points are *constant solutions* of S. An autonomous nonlinear system can have either no equilibrium points, one equilibrium point, or multiple equilibrium points, as can be seen from the following example.

Example 7-3-1: Equilibrium Points

Consider the following one-dimensional nonlinear system S:

$$\dot{x} = u - (x - 1)^2$$

Suppose $u(t) = r$ for $t \geq 0$ for some $r \in \mathbf{R}$. Setting $g(x, r) = 0$ and solving for x yields:

$$\hat{x} = 1 \pm (r)^{1/2}$$

Since x must be real, it follows that when $r < 0$ there are no equilibrium points. When $r = 0$, there is a single equilibrium point at $\hat{x} = 1$. Finally, when $r > 0$, there are a pair of equilibrium points at $\hat{x} = 1 \pm (r)^{1/2}$.

Although equilibrium point analysis is applicable to nonlinear systems in general, our principal interest is in one particular nonlinear system, the dynamic model of a robotic manipulator. The following result summarizes the equilibrium points of a robotic arm which arise when constant torques are generated by the joint actuators.

Proposition 7-3-1: Equilibrium Point. Suppose the applied torque vector of a robotic arm is constant and equal to $\tau(t) = r$ for some $r \in \mathbf{R}^n$. Then $\hat{x}^T = [\hat{q}^T, \hat{v}^T]$ is an equilibrium point of the arm if and only if:

$$h(\hat{q}) = r$$

$$\hat{v} = 0$$

Proof. From Prop. 7-2-1, the general state equation of a robotic arm is:

$$\dot{q} = v$$

$$\dot{v} = D^{-1}(q)[\tau - h(q) - c(q, v) - b(v)]$$

If $\tau(t) = r$ for $t \geq 0$, then, setting the right-hand side equal to zero and solving for q and v, we get $v = 0$ and:

$$D^{-1}(q)[r - h(q) - c(q, 0) - b(0)] = 0$$

Multiplying both sides by the manipulator inertia tensor $D(q)$ and recalling Eqs. (6-3-3) and (6-4-8) then yields:

$$\begin{aligned} h(q) &= r - c(q, 0) - b(0) \\ &= r - c(q, 0) \\ &= r \end{aligned}$$

Notice that the equilibrium points of a robotic arm associated with a constant input are determined exclusively by the *gravity loading* term $h(q)$. The accompanying constraint, $v = 0$, simply states that the robotic arm is motionless at the points

of equilibrium, since v represents the vector of joint velocities. Depending upon the value of the applied torque, the joint position constraint in Prop. 7-3-1 can have either no solution, a unique solution, or multiple solutions. The following examples illustrate this point.

Example 7-3-2: One-Axis Robot

Consider the one-axis robot or inverted pendulum shown in Fig. 7-2. The dynamic model of this robot was computed in Eq. (7-2-1), where the gravity loading term can be identified by the coefficient g_0. From Prop. 7-3-1, the equilibrium points are at $v = 0$ and $h(q) = r$, where $h(q) = g_0(m_1/2 + m_L)a_1C_1$. Recalling that $C_1 = \cos q_1$, the equilibrium positions of the arm are:

$$\hat{q} = \pm\arccos\frac{r}{g_0(m_1/2 + m_L)a_1} \tag{7-3-3}$$

Clearly, equilibrium points exist if and only if $|r| \leq g_0(m_1/2 + m_L)a_1$. For example, if the applied torque is $r = g_0(m_1/2 + m_L)a_1/2$, then there are two equilibrium points at $\hat{x} = [\pm\pi/3, 0]^T$.

Example 7-3-3: Two-Axis Planar Articulated Robot

Consider the two-axis planar articulated robot shown in Fig. 7-3. The dynamic model of this robot was computed in Eq. (7-2-4), where the gravity loading term again contains coefficient g_0:

$$h(q) = g_0\left[\left(\frac{m_1}{2} + m_2\right)a_1C_1 + \frac{m_2a_2C_{12}}{2}, \frac{m_2a_2C_{12}}{2}\right]^T \tag{7-3-4}$$

From Prop. 7-3-1, the equilibrium points are at $v = 0$ and $h(q) = r$. Note from Eq. (7-3-4) that if q is an equilibrium position, then $h_2(q) = g_0m_2a_2C_{12}/2 = r_2$. But this implies $h_1(q) = g_0(m_1/2 + m_2)a_1C_1 + r_2 = r_1$. Solving for q_1 and q_2 yields:

$$\hat{q}_1 = \pm\arccos\frac{r_1 - r_2}{g_0(m_1/2 + m_2)a_1} \tag{7-3-5a}$$

$$\hat{q}_2 = \pm\arccos\left(\frac{2r_2}{g_0m_2a_2}\right) - \hat{q}_1 \tag{7-3-5b}$$

Whereas the one-axis robot in Eq. (7-3-3) has two equilibrium points, the two-axis planar articulated robot in Eq. (7-3-5) has four equilibrium points. The simplest special case occurs when the applied torque is $r = 0$. In this case the four equilibrium positions of the arm are:

$$\hat{q}^1 = \left[\frac{\pi}{2}, 0\right]^T \tag{7-3-6a}$$

$$\hat{q}^2 = \left[\frac{\pi}{2}, \pi\right]^T \tag{7-3-6b}$$

$$\hat{q}^3 = \left[-\frac{\pi}{2}, \pi\right]^T \tag{7-3-6c}$$

$$\hat{q}^4 = \left[-\frac{\pi}{2}, 0\right]^T \tag{7-3-6d}$$

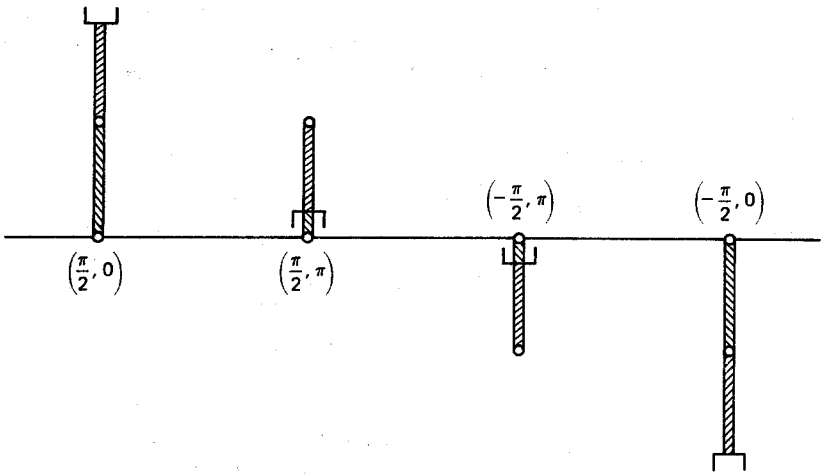


Figure 7-5 Equilibrium positions of two-axis robot when $r = 0$.

These four solutions are shown in Fig. 7-5. Since $|\hat{q}_2| = k\pi$ for $k = 0, 1$, the two links are always aligned, and since $|\hat{q}_2| = \pi/2$, the arm is pointing either straight up or straight down.

Exercise 7-3-1: Three-Axis SCARA Robot. Consider the three-axis SCARA robot in Fig. 7-4 whose dynamic model is given in Eq. (7-2-10). Suppose $\tau(t) = r$ for $t \geq 0$. For what values of $r \in \mathbf{R}^3$ do equilibrium points exist?

Now that a method is available for determining the equilibrium points or constant solutions of a robotic arm, we next examine the properties of those solutions. Because an equilibrium point of a nonlinear system is a constant solution, any solution which starts at an equilibrium point will stay there. However, what happens when a solution starts out *near* an equilibrium point $\hat{x}$ in the sense that $\|x(0) - \hat{x}\|$ is small? Does the solution $x(t)$ converge to $\hat{x}$ as t approaches infinity? To examine this question, we first introduce an important qualitative concept, the notion of asymptotic stability.

Definition 7-3-2: Asymptotic Stability. An equilibrium point $\hat{x}$ of the system S in Eq. (7-3-1) is *asymptotically stable* if and only if for each $\epsilon > 0$ there exists a $\delta > 0$ such that if $\|x(0) - \hat{x}\| < \delta$, then $\|x(t) - \hat{x}\| < \epsilon$ for $t \geq 0$ and:

$$x(t) \to \hat{x} \qquad \text{as} \qquad t \to \infty$$

Thus if an equilibrium point $\hat{x}$ is asymptotically stable, then any solution which starts out sufficiently close to $\hat{x}$ stays close in the sense that $\|x(t) - \hat{x}\|$ remains small and, in addition, the solution asymptotically approaches $\hat{x}$ in the limit as $t \to \infty$. The notion of asymptotic stability can be interpreted geometrically with the aid of the following set, called an *open ball*.

Definition 7-3-3: Open Ball. An *open ball* in $\mathbf{R}^m$ of radius ρ centered at $\hat{x}$ is denoted $B(\hat{x}, \rho)$ and is defined:

$$B(\hat{x}, \rho) \triangleq \{x \in \mathbf{R}^m: \|x - \hat{x}\| < \rho\}$$

Hence the open ball $B(\hat{x}, \rho)$ is the set of all vectors x in $\mathbf{R}^m$ whose distance from the vector $\hat{x}$, as measured by the norm, is less than ρ. Asymptotic stability has a simple interpretation in terms of open balls, as can be seen in Fig. 7-6, which illustrates the case $n = 2$. Here, if $\hat{x}$ is asymptotically stable, then for each $\epsilon > 0$ we can always find a $\delta > 0$ such that if a solution starts inside the ball $B(\hat{x}, \delta)$ at time $t = 0$, it will remain inside the ball $B(\hat{x}, \epsilon)$ for $t \geq 0$ and will approach $\hat{x}$ in the limit as $t \to \infty$.

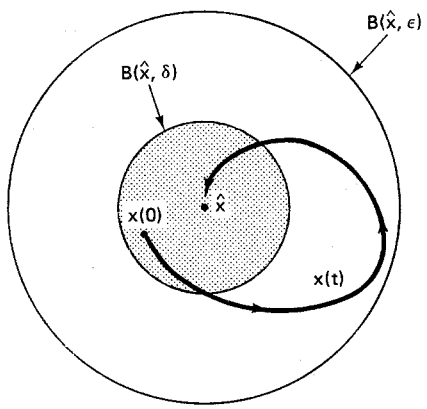


Figure 7-6 Asymptotic stability of $\hat{x}$.

7-3-1 Liapunov's First Method

There is a simple sufficient condition that can be used to test an equilibrium point for asymptotic stability. To see this, we again consider the notion of a Jacobian matrix.

Definition 7-3-4: Jacobian Matrix. Let g: $\mathbf{R}^n \to \mathbf{R}^m$ be differentiable. Then the *Jacobian matrix of* g, evaluated at x, is the following $m \times n$ matrix $G(x)$ of partial derivatives of g with respect to x:

$$G_{kj}(x) \triangleq \frac{\partial}{\partial x_j} g_k(x) \qquad 1 \leq k \leq m,\ 1 \leq j \leq n$$

Thus the component of the Jacobian matrix of g in the kth row and jth column is the partial derivative of the kth component of $g(x)$ with respect to the jth component of x. If g: $\mathbf{R}^n \to \mathbf{R}$, then the Jacobian matrix reduces to a $1 \times n$ row vector called the *gradient of* g. More generally, the kth row of $G(x)$ is the gradient of the scalar-valued function $g_k(x)$. The Jacobian matrix can now be used to formulate a simple sufficient condition which ensures that an equilibrium point of a nonlinear system S is asymptotically stable. In particular, we have the following result, which is known as *Liapunov's first method* (Vidyasagar, 1978).

Proposition 7-3-2: Liapunov's First Method. Let $\hat{x}$ be an equilibrium point of the system S in Eq. (7-3-1) associated with a constant input $u(t) \equiv r$, and let

$G(\hat{x})$ be the Jacobian matrix of $g(x, r)$ evaluated at $x = \hat{x}$. Finally, let λ_k denote the kth eigenvalue of $G(\hat{x})$ for $1 \leq k \leq m$. Then $\hat{x}$ is asymptotically stable if the real part of each eigenvalue is negative. That is,

$$\text{Re}\,(\lambda_k) < 0 \qquad 1 \leq k \leq m$$

Liapunov's first method tells us that if all of the eigenvalues of the Jacobian matrix $G(\hat{x})$ lie in the open left half of the complex plane, then the equilibrium point $\hat{x}$ is asymptotically stable. Note that Prop. 7-3-1 is only a *sufficient* condition for asymptotic stability; it is not a necessary condition.

Liapunov's first method, sometimes called his *indirect* method, can be interpreted as a technique in which the stability of an equilibrium point of a nonlinear system S is inferred by examining the stability properties of an associated linear system. It is in this sense that the method is indirect. To generate the associated linear system, we define $\delta x(t) \triangleq x(t) - \hat{x}$ as the *variation of* $x(t)$ about the equilibrium point $\hat{x}$. If we perform a Taylor series expansion of $g(x, r)$ about the equilibrium point $x = \hat{x}$ and drop the higher-order terms, this yields:

$$\begin{aligned}
\delta\dot{x} &= \dot{x} \\
&= g(x, r) \\
&= g(\hat{x}, r) + G(\hat{x})\,\delta x + \cdots \\
&\approx g(\hat{x}, r) + G(\hat{x})\,\delta x \\
&= G(\hat{x})\,\delta x
\end{aligned} \tag{7-3-7}$$

We refer to the linear system in Eq. (7-3-7) as the *linearization* of the nonlinear system S about the operating point $(x, u) = (\hat{x}, r)$. The origin of the linear system in Eq. (7-3-7) is asymptotically stable if and only if the eigenvalues of $G(\hat{x})$ all lie in the open left half of the complex plane (Ogata, 1970). Consequently, Prop. 7-3-2 can be interpreted as saying that the stability properties of the nonlinear system S, in the neighborhood of the equilibrium point $\hat{x}$, can be inferred from the stability properties of the associated linearized system in Eq. (7-3-7).

Example 7-3-4: Asymptotic Stability

Consider the following two-dimensional nonlinear system S:

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= u(1 - x_2) - x_1
\end{aligned}$$

Suppose $u(t) = r$ for $t \geq 0$. Setting $\dot{x} = 0$ and solving for x yields $x_2 = 0$ and $x_1 = r$. Thus this system has a single equilibrium point at:

$$\hat{x} = [r, 0]^T$$

The Jacobian matrix $G(x)$ for this system is the following 2×2 matrix of partial derivatives:

$$G(x) = \begin{bmatrix} 0 & 1 \\ -1 & -r \end{bmatrix}$$

Note that the Jacobian matrix, in this case, does not depend explicitly on x, although it does depend upon the constant input $u(t) = r$. Evaluating the characteristic polynomial of $G(\hat{x})$, we get:

$$\begin{aligned}\Delta(\lambda) &= \det\{\lambda I - G(\hat{x})\} \\ &= \lambda(\lambda + r) + 1 \\ &= \lambda^2 + r\lambda + 1 \\ &= (\lambda - \lambda_1)(\lambda - \lambda_2)\end{aligned}$$

Here the eigenvalues of $G(\hat{x})$ are:

$$\lambda_{1,2} = \frac{-r \pm (r^2 - 4)^{1/2}}{2}$$

Applying Prop. 7-3-1, we can make the following claim about the stability of the equilibrium point $\hat{x} = [r, 0]^T$. If $r > 0$, then both eigenvalues of $G(\hat{x})$ have negative real parts. Consequently, the following locus of equilibrium points is asymptotically stable:

$$\hat{x} = [r, 0]^T \qquad r > 0$$

Liapunov's first method can now be applied to analyze the stability properties of the equilibrium points or constant solutions of a robotic arm.

Example 7-3-5: One-Axis Robot

Consider the one-axis robot, or inverted pendulum, shown in Fig. 7-2. To simplify the analysis, we assume that the friction is purely viscous friction. If we define $\delta \triangleq (m_1/3 + m_L)a_1^2$ and let $\tau_1(t) = r$ for $t \geq 0$, then from Eqs. (7-2-2) and (6-3-3) the state equations of the arm are:

$$\begin{aligned}\dot{q} &= v \\ \dot{v} &= \frac{r - g_0(m_1/2 + m_L)a_1C_1 - b_1^v v}{\delta}\end{aligned}$$

To examine the stability of the equilibrium points of this system, we first compute the Jacobian matrix. Differentiating the right-hand side of the state equations yields:

$$G(x) = \begin{bmatrix} 0 & 1 \\ \dfrac{g_0(m_1/2 + m_L)a_1S_1}{\delta} & -\dfrac{b_1^v}{\delta} \end{bmatrix}$$

The Jacobian matrix can now be evaluated at the equilibrium points, and the eigenvalues computed. The characteristic polynomial of $G(\hat{x})$ is:

$$\begin{aligned}\Delta(\lambda) &= \det\{\lambda I - G(\hat{x})\} \\ &= \lambda\left(\lambda + \frac{b_1^v}{\delta}\right) - \frac{g_0(m_1/2 + m_L)a_1S_1}{\delta} \\ &= \lambda^2 + \left(\frac{b_1^v}{\delta}\right)\lambda - \frac{g_0(m_1/2 + m_L)a_1S_1}{\delta} \\ &= (\lambda - \lambda_1)(\lambda - \lambda_2)\end{aligned}$$

Here the two eigenvalues are:

$$\lambda_{1,2} = \frac{-\frac{b_1^v}{\delta} \pm \left[\left(\frac{b_1^v}{\delta}\right)^2 + \frac{2g_0(m_1 + 2m_L)a_1S_1}{\delta}\right]^{1/2}}{2}$$

It is clear that for those equilibrium points for which $S_1 = \sin(q_1) < 0$, both eigenvalues are in the open left half of the complex plane. Thus equilibrium points in the range $-\pi < q_1 < 0$ are asymptotically stable. Recall from Example 7-3-2 that if $|r| < g_0(m_1/2 + m_L)a_1$, then the one-axis robot has a pair of equilibrium points located symmetrically about the horizontal x^0 axis, as shown in Fig. 7-7. We conclude that the solution below the x^0 axis is asymptotically stable.

The asymptotic stability of the lower equilibrium position in Fig. 7-7 can be confirmed physically if one "perturbs" the pendulum a small distance away from the equilibrium position. Note that $dC_1/dq_1 > 0$ when $-\pi/2 < q_1 < 0$. If the pendulum is lifted up slightly, then the torque contribution due to gravity, $-g_0(m_1/2 + m_L)a_1C_1$, becomes more negative, which causes a downward restoring torque. Similarly, if the pendulum is pulled down slightly, then the torque due to gravity becomes less negative, which causes an upward restoring torque. Hence the lower equilibrium position is indeed asymptotically stable.

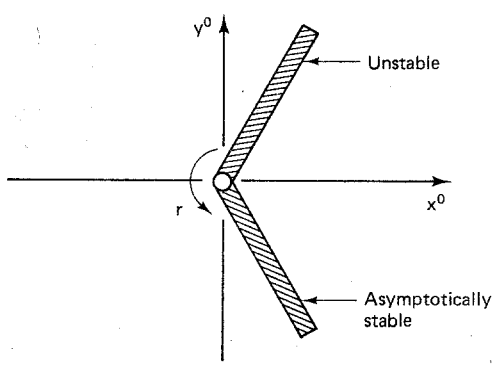


Figure 7-7 Stable and unstable equilibrium points of a one-axis robot.

7-3-2 Liapunov's Second Method

Each asymptotically stable equilibrium point has an open region surrounding it called a *domain of attraction*. As the name implies, the domain of attraction is a set Ω with the following property:

$$x(0) \in \Omega \quad \Rightarrow \quad x(t) \rightarrow \hat{x} \quad \text{as} \quad t \rightarrow \infty \tag{7-3-8}$$

That is, any solution which starts out inside the domain of attraction Ω will be attracted to the equilibrium point $\hat{x}$ in the limit as $t \rightarrow \infty$. There is a relatively simple sufficient condition, called *Liapunov's second method*, that can be used to both establish the asymptotic stability of an equilibrium point and *estimate* its domain of attraction. To formulate this condition, we first consider the following class of functions, called *Liapunov functions*.

Definition 7-3-5: Liapunov Function. Let Ω be an open region in $\mathbf{R}^m$ containing the origin. A function V_L: $\Omega \rightarrow \mathbf{R}$ is a *Liapunov function* on Ω if and only if:

1. $V_L(x)$ has a continuous derivative.
2. $V_L(0) = 0$.
3. $V_L(x) > 0$ for $x \neq 0$.

We refer to conditions 2 and 3 collectively by saying that the function $V_L(x)$ is *positive-definite*. Thus a Liapunov function is a continuously differentiable positive-definite function of the state. Perhaps the simplest example of a Liapunov function is $V_L(x) = \|x\|^2 = x^T x$. More generally, if W is any symmetric weighting matrix with positive eigenvalues, then the following is a valid Liapunov function:

$$V_L(x) = x^T W x \tag{7-3-9}$$

Often we take W to be a diagonal matrix with positive diagonal elements. For example, if $W = I$, then $V_L(x)$ reduces to $\|x\|^2$. Another class of Liapunov functions consists of a weighted sum of even powers of the components of x. In this case the weighting coefficients must all be positive.

To simplify the analysis, we will assume that the equilibrium point of interest is located at the origin. If this is not the case, we can always first perform a change of variables to translate it to the origin. To verify the asymptotic stability of $\hat{x} = 0$, we evaluate $V_L(x(t))$ along solutions of the system S. If we can demonstrate that $V_L(x(t))$ decreases along solutions of S, then it must be the case that $V_L(x(t)) \rightarrow 0$ as $t \rightarrow \infty$. But since $V_L(x)$ is a Liapunov function, this implies that $x(t) \rightarrow 0$ as $t \rightarrow \infty$. This is the essential idea behind Liapunov's second method.

Proposition 7-3-3: Liapunov's Second Method. Let $\hat{x} = 0$ be an equilibrium point of the system S in Eq. (7-3-1) associated with a constant input $u(t) = r$. Next, let V_L: $\Omega_\rho \rightarrow \mathbf{R}$ be a Liapunov function on Ω_ρ where $\Omega_\rho = \{x: V_L(x) < \rho\}$ for some $\rho > 0$. Then $\hat{x}$ is asymptotically stable with domain of attraction Ω_ρ if, along solutions of S:

1. $\dot{V}_L(x(t)) \leq 0$
2. $\dot{V}_L(x(t)) \equiv 0 \Rightarrow x(t) \equiv 0$

Note that condition 1 in Prop. 7-3-3 ensures that the value of $V_L(x(t))$ does not increase along solutions of the system S. The $\equiv$ sign in condition 2 denotes "identically equal," which means for all t. Therefore condition 2 says that there are no solutions of S for which $V_L(x(t))$ remains constant except the equilibrium point solution, $x(t) = 0$. But if $V_L(x(t))$ does not increase and also does not stay constant, then it must decrease. Therefore $V(x(t)) \rightarrow 0$ as $t \rightarrow \infty$, which means that $x(t) \rightarrow 0$ as $t \rightarrow \infty$. The beauty of Liapunov's second method, which is also called his *direct* method, is that conditions 1 and 2 can be checked without *solving* the nonlinear system. Indeed from Eq. (7-3-1), the time derivative of $V_L(x(t))$, evaluated along solutions of S, is:

$$\dot{V}_L(x) = \left[\frac{\partial V_L(x)}{\partial x}\right]\dot{x}$$

$$= \left[\frac{\partial V_L(x)}{\partial x}\right] g(x, r) \tag{7-3-10}$$

Thus $\dot{V}_L(x)$ is simply the gradient of $V_L(x)$ times the right-hand side, $g(x, r)$. There is no need to solve the nonlinear system. This is important because, in most cases, a closed-form *expression* for the solution of Eq. (7-3-1) does not exist.

We illustrate Liapunov's second method with two examples.

Example 7-3-6: Domain of Attraction

Consider the following two-dimensional nonlinear system:

$$\dot{x}_1 = x_2 \tag{7-3-11a}$$

$$\dot{x}_2 = u - x_1 - x_2 + x_2^3 \tag{7-3-11b}$$

Suppose $u(t) = r$. Then setting $\dot{x} = 0$ and solving for x yields $x_2 = 0$ and $x_1 = r$. Hence there is a single equilibrium point at:

$$\hat{x} = [r, 0]^T \tag{7-3-12}$$

For convenience, suppose $r = 0$, which implies $\hat{x} = 0$. As a candidate Liapunov function, try:

$$V_L(x) = \alpha x_1^2 + \beta x_2^2 \qquad \alpha > 0, \beta > 0 \tag{7-3-13}$$

Note that $V_L(x)$ is of the same form as Eq. (7-3-9) where the weighting matrix is $W = \text{diag}\,\{\alpha, \beta\}$. Evaluating $\dot{V}_L(x)$ along solutions of Eq. (7-3-11) yields:

$$\begin{aligned}\dot{V}_L(x) &= \left[\frac{\partial V_L(x)}{\partial x}\right]\dot{x} \\ &= [2\alpha x_1, 2\beta x_2]\dot{x} \\ &= [2\alpha x_1, 2\beta x_2][x_2, -(x_1 + x_2 - x_2^3)]^T \\ &= 2\alpha x_1 x_2 - 2\beta x_2(x_1 + x_2 - x_2^3) \\ &= 2(\alpha - \beta)x_1 x_2 - 2\beta x_2^2(1 - x_2^2)\end{aligned} \tag{7-3-14}$$

Since the factor $x_1 x_2$ changes sign at the equilibrium point $x = 0$, we must constrain α and β so as to eliminate it. That is, if $\beta = \alpha$, then:

$$\dot{V}_L(x) = -2\beta x_2^2(1 - x_2^2) \qquad \beta = \alpha \tag{7-3-15}$$

We see from Eq. (7-3-15) that $\dot{V}_L(x) \leq 0$ along solutions of Eq. (7-3-11) if $\beta = \alpha$ and if x is constrained to the following open region containing the equilibrium point:

$$|x_2| < 1 \tag{7-3-16}$$

Thus condition 1 of Liapunov's second method is satisfied. To check condition 2 we use Eqs. (7-3-11) and (7-3-15) as follows:

$$\dot{V}_L(x(t)) \equiv 0 \Rightarrow x_2(t) \equiv 0$$

$$\Rightarrow \dot{x}_2(t) \equiv 0$$

$$\Rightarrow \quad x_1(t) \equiv 0 \tag{7-3-17}$$

Hence $\dot{V}_L(x(t)) \equiv 0 \Rightarrow x(t) \equiv 0$. That is, the only solution of Eq. (7-3-11) for which $V_L(x(t))$ is constant is the equilibrium solution, $x(t) = 0$. Therefore, from Prop. 7-3-3, $\hat{x} = 0$ is asymptotically stable. To estimate the domain of attraction, we must find the largest $\rho > 0$ such that Ω_ρ in Prop. 7-3-3 does not violate constraint (7-3-16). Now:

$$\begin{aligned} \Omega_\rho &= \{x: V_L(x) < \rho\} \\ &= \{x: \alpha x_1^2 + \beta x_2^2 < \rho\} \\ &= \{x: \alpha(x_1^2 + x_2^2) < \rho\} \end{aligned} \tag{7-3-18}$$

If we choose $\rho = \alpha$, then $|x_2| < 1$ as required by constraint (7-3-16). In this case the domain of attraction Ω_α is the ball of unit radius shown in Fig. 7-8. Any solution starting out inside $\Omega_\alpha = B(0, 1)$ is guaranteed to approach $\hat{x} = 0$ in the limit as t approaches infinity.

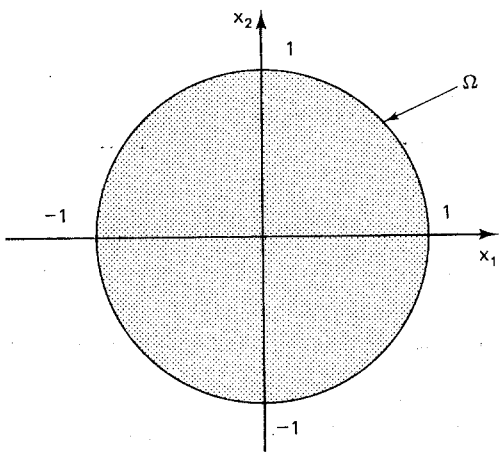


Figure 7-8 Domain of attraction for example 7-3-6.

Example 7-3-6 illustrates both the power of Liapunov's second method and its principal weakness. The power is clear from the fact that we were able to show that the domain of attraction of $\hat{x} = 0$ includes the ball $B(0, 1)$ without ever solving the system. However, to demonstrate this we had to choose the "right" Liapunov function. Unfortunately, there is no general systematic procedure that is guaranteed to produce a suitable Liapunov function. Instead, experience and intuition must be used to generate appropriate Liapunov function candidates. Often the general formulation in Eq. (7-3-9) will work for a suitable choice of W, but in some cases an alternative form is needed, as can be seen from the following example:

Example 7-3-7: One-Axis Robot

Consider the one-axis robot or inverted pendulum shown in Fig. 7-2. If we define $\delta \triangleq (m_1/3 + m_L)a_1^2$, then from Eq. (7-2-2) the state equation for a constant input $u(t) = r$ is:

$$\dot{q} = v \tag{7-3-19a}$$

$$\dot{v} = \frac{r - g_0(m_1/2 + m_L)a_1 C_1 - b_1(v)}{\delta} \tag{7-3-19b}$$

If $|r| < g_0(m_1/2 + m_L)a_1$, then from Example 7-3-2, the system has two equilibrium points occurring at:

$$\hat{x} = \left[\pm \arccos \frac{r}{g_0(m_1/2 + m_L)a_1}, 0\right]^T \tag{7-3-20}$$

Using Liapunov's first method in Example 7-3-5, we were able to show that the equilibrium point associated with the minus sign is asymptotically stable (for the case $b_1(v) = b_1^v(v)$). Let us now use Liapunov's second method to estimate its domain of attraction. For convenience, we assume $r = 0$, which puts the equilibrium point at $\hat{x} = [-\pi/2, 0]^T$, corresponding to the arm hanging straight down. First we perform a change of variable to translate this equilibrium point to the origin. If $z \triangleq [q + \pi/2, v]^T$, then the new state equation in term of z is:

$$\dot{z}_1 = z_2 \tag{7-3-21a}$$

$$\begin{aligned}
\dot{z}_2 &= \dot{v} \\
&= -\frac{g_0(m_1/2 + m_L)a_1 C_1 + b_1(v)}{\delta} \\
&= -\frac{g_0(m_1/2 + m_L)a_1 \cos q + b_1(z_2)}{\delta} \\
&= -\frac{g_0(m_1/2 + m_L)a_1 \cos (z_1 - \pi/2) + b_1(z_2)}{\delta} \\
&= -\frac{g_0(m_1/2 + m_L)a_1 \sin z_1 + b_1(z_2)}{\delta}
\end{aligned} \tag{7-3-21b}$$

Note that $\hat{z} = 0$ is now an equilibrium point of the transformed system. To find a Liapunov function which decreases along solutions of this system, we must somehow cancel the term containing $\sin z_1$, since this changes sign at the equilibrium point. Therefore, we must include a term in $V_L(z)$ whose derivative is proportional to $\sin z_1$. As a candidate Liapunov function, try:

$$V_L(z) = \alpha(1 - \cos z_1) + \beta z_2^2 \qquad \alpha > 0, \beta > 0 \tag{7-3-22}$$

Note that $V_L(z)$ is continuously differentiable, $V_L(0) = 0$, and $V_L(z) > 0$ if $0 < |z_1| < \pi$. Thus, from Def. 7-3-5, $V_L(z)$ is a valid Liapunov function if z is constrained to the following open region which contains the equilibrium point:

$$|z_1| < \pi \tag{7-3-23}$$

Evaluating $\dot{V}_L(z(t))$ along solutions of Eq. (7-3-21) yields:

$$\begin{aligned}
\dot{V}_L(z) &= \left[\frac{\partial V_L(z)}{\partial z}\right]\dot{z} \\
&= [\alpha \sin z_1, 2\beta z_2]\dot{z} \\
&= [\alpha \sin z_1, 2\beta z_2]\left[z_2, -\frac{g_0(m_1/2 + m_L)a_1 \sin z_1 + b_1(z_2)}{\delta}\right]^T \\
&= \alpha (\sin z_1) z_2 - 2\beta z_2 \frac{g_0(m_1/2 + m_L)a_1 \sin z_1 + b_1(z_2)}{\delta}
\end{aligned}$$

$$= \left[\alpha - \frac{\beta g_0(m_1 + 2m_L)a_1}{\delta}\right](\sin z_1)\, z_2$$

$$- \frac{2\beta(b_1^v z_2^2 + |z_2|(b_1^d + (b_1^s - b_1^d)\exp(-|z_2|/\epsilon)))}{\delta} \tag{7-3-24}$$

The term containing $(\sin z_1)z_2$ changes sign at $z = 0$ and therefore must be eliminated through a judicious choice of α and β. That is, if $\beta = \alpha\delta/[g_0(m_1 + 2m_L)a_1]$, then:

$$\dot{V}_L(z) = -\frac{2\beta(b_1^v z_2^2 + |z_2|(b_1^d + (b_1^s - b_1^d)\exp(-|z_2|/\epsilon)))}{\delta} \qquad \beta = \frac{\alpha\delta}{g_0(m_1 + 2m_L)a_1} \tag{7-3-25}$$

Therefore $\dot{V}_L(z) \leq 0$ when $\beta = \alpha\delta/[g_0(m_1 + 2m_L)a_1]$. Hence condition 1 of Liapunov's second method is satisfied. To check condition 2, we have, from Eqs. (7-3-21) and (7-3-25):

$$\begin{aligned} \dot{V}_L(z(t)) \equiv 0 &\Rightarrow z_2(t) \equiv 0 \\ &\Rightarrow \dot{z}_2(t) \equiv 0 \\ &\Rightarrow \sin z_1(t) \equiv 0 \\ &\Rightarrow z_1(t) \equiv k\pi \end{aligned} \tag{7-3-26}$$

where k is an integer. But we already have the restriction $|z_1| < \pi$, from inequality (7-3-23). Thus $k = 0$, which means that $\dot{V}_L(z(t)) \equiv 0$ implies that $z(t) \equiv 0$. Hence condition 2 is satisfied and $\hat{z} = 0$ is indeed asymptotically stable. To estimate the domain of attraction, we must find the largest $\rho > 0$ such that Ω_ρ in Prop. 7-3-3 does not violate constraint (7-3-23). Now:

$$\begin{aligned} \Omega_\rho &= \{z: V_L(z) < \rho\} \\ &= \{z: \alpha(1 - \cos z_1) + \beta z_2^2 < \rho\} \\ &= \left\{z: \alpha\left[1 - \cos z_1 + \frac{\delta z_2^2}{g_0(m_1 + 2m_L)a_1}\right] < \rho\right\} \end{aligned} \tag{7-3-27}$$

If we chose $\rho = 2\alpha$, then this guarantees that $|z_1| < \pi$, as required in constraint (7-3-23). In this case, the largest possible value for $|z_2|$ is $\gamma \triangleq [2g_0(m_1 + 2m_L)a_1/\delta]^{1/2}$. Thus the domain of attraction is the oval-shaped region $\Omega_{2\alpha}$ shown in Fig. 7-9.

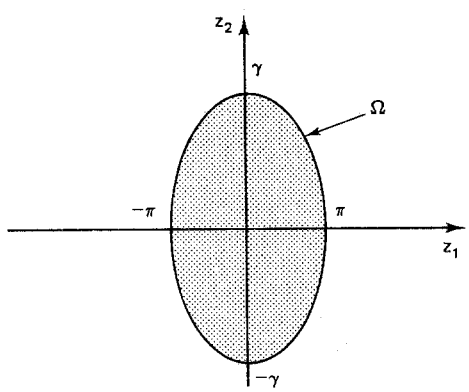


Figure 7-9 Domain of attraction for one-axis robot.

The domain of attraction in Fig. 7-9 has a simple physical interpretation. If the arm starts out motionless, then any initial position satisfying $|z_1(0)| < \pi$ will cause the arm or pendulum to exhibit a damped oscillation which settles in the stable straight-down position, $z_1 = 0$. Similarly, if the initial position is straight down and the initial angular velocity satisfies $|z_2(0)| < \gamma$, then the arm will again settle in the stable straight-down equilibrium position, $z_1(t) = 0$. However, if the initial state is such that $z(0)$ lies outside the set $\Omega_{2\alpha}$, then the equilibrium point $\hat{z} = 0$ may fail to "attract" the solution in the limit as time approaches infinity. For example, if the arm were initially moving with a large angular velocity, it might wrap around one or more revolutions before coming to rest at $\hat{z} = [2\pi k, 0]^T$ for some nonzero integer k.

7-4 LINEAR FEEDBACK SYSTEMS

A robotic arm is a highly nonlinear system. However, to control a robotic arm, it is still helpful to make use of techniques developed for linear systems. Indeed, many commercial robot controllers model the arm as if it were a linear system with a time-varying load or disturbance. To present this approach to robot control, we briefly review some relevant concepts from linear feedback theory (Ogata, 1970; Kuo, 1982).

7-4-1 Transfer Function

Many physical systems can be effectively modeled by a suitable mth-order linear differential equation with constant coefficients. If $u(t)$ represents the input and $y(t)$ the output, then the general model can be expressed as follows:

$$\sum_{k=0}^{m} a_k y^{(k)}(t) = \sum_{k=0}^{m} b_k u^{(k)}(t) \tag{7-4-1}$$

Here $u^{(k)}(t)$ and $y^{(k)}(t)$ denote the kth derivative of $u(t)$ and $y(t)$, respectively. Without loss of generality, we can assume that Eq. (7-4-1) has been *normalized* so that $a_m = 1$. If we take the Laplace transform of both sides of Eq. (7-4-1), assuming zero conditions at time $t = 0^-$, then Eq. (7-4-1) reduces to:

$$\left(s^m + \sum_{k=0}^{m-1} a_k s^k\right) Y(s) = \left(\sum_{k=0}^{m} b_k s^k\right) U(s) \tag{7-4-2}$$

Here $U(s) = \mathcal{L}\{u(t)\}$ and $Y(s) = \mathcal{L}\{y(t)\}$ denote the Laplace transforms of $u(t)$ and $y(t)$, respectively. We refer to the ratio $G(s) \triangleq Y(s)/U(s)$ as the *transfer function* of the linear system in Eq. (7-4-1). From Eq. (7-4-2) we can solve for the transfer function as follows:

$$G(s) = \frac{\sum_{k=0}^{m} b_k s^k}{s^m + \sum_{k=0}^{m-1} a_k s^k} \tag{7-4-3}$$

The transfer function is a compact algebraic representation of the input/output behavior of the linear system in Eq. (7-4-1). Once the transfer function is known, computing the output of the system for any input is straightforward because, by definition, $Y(s) = G(s)U(s)$, as shown in Fig. 7-10. The representation of the system in Fig. 7-10 is referred to as a *block diagram* description of the system.

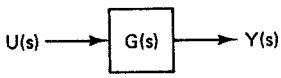


Figure 7-10 Block diagram of a system with transfer function $G(s)$.

Example 7-4-1: Transfer Function

Consider a linear system described by the following second-order constant-coefficient differential equation:

$$2y^{(2)}(t) + 3y^{(1)}(t) + 4y(t) = 6u^{(1)}(t) + 2u(t)$$

There is no need to actually take Laplace transforms and then factor out the $U(s)$ and $Y(s)$. Instead, we can write down the transfer function directly from inspection of the coefficients as:

$$G(s) = \frac{6s + 2}{(2s^2 + 3s + 4)}$$

$$= \frac{3s + 1}{s^2 + \frac{3}{2}s + 2}$$

Given an input $u(t)$, to determine $y(t)$ it is clear from Fig. 7-10 that all we have to do is take the inverse Laplace transform of $Y(s) = G(s)U(s)$. Of course, this necessitates taking Laplace transforms and inverse Laplace transforms of a variety of signals. A list of common signals and their Laplace transforms is provided in Table 7-1, where $\delta(t)$ denotes the unit impulse and $1(t)$ denotes the unit step.

TABLE 7-1 LAPLACE TRANSFORMS OF COMMON SIGNALS

$f(t)$	$F(s)$
$\delta(t)$	1
$\frac{t^n 1(t)}{n!}$	$\frac{1}{s^{n+1}}$
$\frac{t^n \exp(-\alpha t)\, 1(t)}{n!}$	$\frac{1}{(s + \alpha)^{n+1}}$
$\exp(-\alpha t)(\sin \omega t)\, 1(t)$	$\frac{\omega}{(s + \alpha)^2 + \omega^2}$
$\exp(-\alpha t)(\cos \omega t)\, 1(t)$	$\frac{s + \alpha}{(s + \alpha)^2 + \omega^2}$

Although the transfer function $G(s)$ is useful as a tool to compute the output $y(t)$ given an input $u(t)$, perhaps the most important feature of the transfer function is that it allows us to draw *qualitative* conclusions about the behavior of the system.

One important qualitative characteristic is system stability. Suppose a *constant* input is applied, namely, $u(t) = r$ for $t \geq 0$. Then $u^{(k)}(t) \equiv 0$ for $k \geq 1$. Recalling that $a_m = 1$, Eq. (7-4-1) then simplifies to:

$$y^{(m)}(t) + \sum_{k=0}^{m-1} a_k y^{(k)}(t) = b_0 r \tag{7-4-4}$$

To analyze the equilibrium points or constant solutions of the linear system in Eq. (7-4-4), we first convert it to the state-space form. Consider the following choice of state variables:

$$x = [y, y^{(1)}, \ldots, y^{m-1}]^T \tag{7-4-5}$$

If we differentiate Eq. (7-4-5) and substitute using Eqs. (7-4-5) and (7-4-4), this results in the following state-space model:

$$\dot{x}_k = x_{k+1} \qquad 1 \leq k < m \tag{7-4-6a}$$

$$\begin{aligned} \dot{x}_m &= y^{(m)} \\ &= b_0 r - \sum_{k=0}^{m-1} a_k y^{(k)} \\ &= b_0 r - \sum_{k=0}^{m-1} a_k x_{k+1} \end{aligned} \tag{7-4-6b}$$

$$y = x_1 \tag{7-4-7}$$

The state and output equations in Eqs. (7-4-6) and (7-4-7) can be recast in more compact vector forms as follows:

$$\dot{x} = Ax + br \tag{7-4-8}$$

$$y = c^T x \tag{7-4-9}$$

Comparing Eqs. (7-4-8) and (7-4-9) with Eqs. (7-4-6) and (7-4-7), we see that $b = [0, 0, \ldots, 0, b_0]^T$, $c = [1, 0, 0, \ldots, 0]^T$, and A is the following $m \times m$ "companion" matrix:

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ \hline -a_0 & -a_1 & -a_2 & \cdots & -a_{m-2} & -a_{m-1} \end{bmatrix} \tag{7-4-10}$$

To obtain the equilibrium points of the linear system in Eq. (7-4-8), we set $\dot{x} = 0$ and solve for x, which yields $A\hat{x} + br = 0$. If $a_0 \neq 0$, then the matrix A in Eq. (7-4-10) is nonsingular and the linear system has a single equilibrium point at:

$$\hat{x} = -A^{-1} br \tag{7-4-11}$$

Whether A is nonsingular or not, if $r = 0$, then it is clear that the linear system

always has an equilibrium point at $\hat{x} = 0$. To analyze the stability of this equilibrium point, we can use Liapunov's first method. Since the system in Eq. (7-4-8) is already linear, the Jacobian matrix is just $G(\hat{x}) = A$. Thus, from Prop. 7-3-2, the origin of the linear system is asymptotically stable if the eigenvalues of A all have negative real parts. In this case we simply say that the linear system itself is stable. Note from Eqs. (7-4-10) and (7-4-3) that the matrix A is determined exclusively by the coefficients of the denominator of the transfer function $G(s)$. The zeros of the denominator polynomial of $G(s)$ are called the *poles* of the linear system. Every pole of $G(s)$ is in fact an eigenvalue of A. This observation leads to the following stability criterion for linear systems (Ogata, 1970):

Proposition 7-4-1: Linear System Stability. A linear system with transfer function $G(s)$ is stable if and only if all of the poles of $G(s)$ lie in the open left half of the complex plane.

Often, transfer functions are used to model components or subsystems of larger systems. In these cases, we need to determine the overall transfer function in terms of the transfer functions of the subsystems. For example, consider the *serial,* or cascade, configuration of two systems $G_1(s)$ and $G_2(s)$ shown in Fig. 7-11. To determine the overall transfer function of this system, we start with the output $Y(s)$ and work backward. That is, $Y(s) = G_2(s)W(s) = G_2(s)G_1(s)U(s)$. Hence:

$$G(s) = G_2(s)G_1(s) \tag{7-4-12}$$

Thus a serial connection of two subsystems is modeled by taking the *product* of the two transfer functions. Clearly, this can be generalized to any number of subsystems in series.

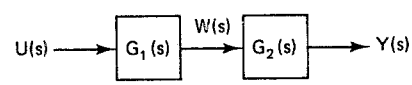


Figure 7-11 Serial configuration of subsystems.

Another common configuration is the *parallel* configuration of two subsystems $G_1(s)$ and $G_2(s)$, shown in Fig. 7-12. Here the + block in Fig. 7-12 denotes a *summer,* whose inputs are added to produce the output. To determine the overall transfer function of the system in Fig. 7-12, we again start with the output $Y(s)$ and work backward. In this case, $Y(s) = G_1(s)U(s) + G_2(s)U(s) = [G_1(s) + G_2(s)]U(s)$. Hence:

$$G(s) = G_1(s) + G_2(s) \tag{7-4-13}$$

Thus a parallel connection of two subsystems is modeled by taking the *sum* of the two transfer functions. Again, this can be generalized to any number of subsystems in parallel.

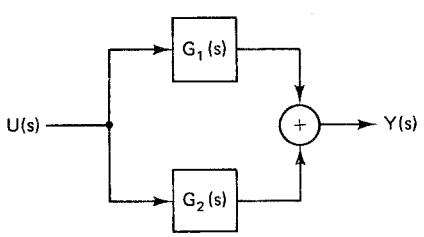


Figure 7-12 Parallel configuration of subsystems.

Perhaps the most important configuration of subsystems for control purposes is the negative feedback configuration of subsystems $G_1(s)$ and $G_2(s)$, shown in Fig. 7-13. Again, a summing junction appears, but in this case its output is the difference of the two inputs as indicated by the plus and minus signs. We refer to $G_1(s)$ as the *open-loop* transfer function, because if the feedback loop is opened by setting $G_2(s) = 0$, the transfer function reduces to $G_1(s)$. The overall transfer function $G(s)$ is called the *closed-loop* transfer function of the feedback system. To develop an expression for the closed-loop transfer function in terms of $G_1(s)$ and $G_2(s)$, we again start with $Y(s)$ and work backward:

$$\begin{aligned} Y(s) &= G_1(s)E(s) \\ &= G_1(s)[U(s) - G_2(s)Y(s)] \\ &= G_1(s)U(s) - G_1(s)G_2(s)Y(s) \end{aligned} \tag{7-4-14}$$

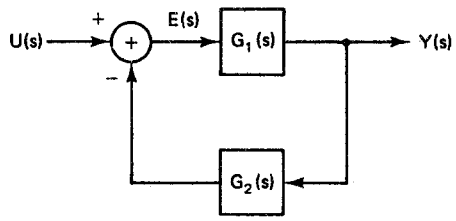


Figure 7-13 Negative feedback configuration of subsystems.

We can now solve Eq. (7-4-14) for the ratio $Y(s)/U(s)$. The resulting closed-loop transfer function of the feedback system is:

$$G(s) = \frac{G_1(s)}{1 + G_1(s)G_2(s)} \tag{7-4-15}$$

Note that $G(s)$ reduces to the open-loop transfer function $G_1(s)$ when we open the feedback loop by setting $G_2(s) = 0$.

It is often the case that $G_2(s) = 1$. When this occurs the system in Fig. 7-13 is called a *unity-gain* feedback system.

Example 7-4-2: Closed-Loop Transfer Function

Consider the feedback configuration in Fig. 7-13, with $G_1(s) = k/[s(Ts + 1)]$ and $G_2(s) = 1$. To find the closed-loop transfer function, we have:

$$\begin{aligned} G(s) &= \frac{G_1(s)}{1 + G_1(s)G_2(s)} \\ &= \frac{k/[s(Ts + 1)]}{1 + k/[s(Ts + 1)]} \\ &= \frac{k}{s(Ts + 1) + k} \\ &= \frac{k}{Ts^2 + s + k} \end{aligned}$$

$$= \frac{k/T}{s^2 + s/T + k/T}$$

To determine whether the feedback system $G(s)$ is stable, from Prop. 7-4-1 we must examine the locations of the poles of $G(s)$. Since the denominator of $G(s)$ is a quadratic, we can factor it directly as:

$$s_{1,2} = \frac{-1 \pm (1 - 4kT)^{1/2}}{2T}$$

If the poles are to reside in the open left half of the complex plane, then we must have $T > 0$ and $k > 0$. Thus the feedback system is stable if and only if $T > 0$ and $k > 0$.

7-4-2 Steady-State Tracking

Unity-gain feedback systems are often used when we want to get the output $y(t)$ to follow or track the input $u(t)$. Note from Fig. 7-13 that if $G_2(s) = 1$, then the output of the summing junction is $E(s) = U(s) - Y(s)$. We refer to the signal $e(t)$, whose Laplace transform is $E(s)$, as the *error* of the system. In control system design, we want to minimize the size of the error for a variety of inputs $u(t)$. One measure of the size of the error is the steady-state tracking error. The *steady-state tracking error* is simply the value of $e(t)$ in the limit as t approaches infinity. To examine the steady-state value of $e(t)$, it is convenient to work with its Laplace transform, $E(s) = \mathcal{L}\{e(t)\}$. From Fig. 7-13 and the definition of $G(s)$, we see that:

$$\begin{aligned} E(s) &= U(s) - Y(s) \\ &= U(s) - G(s)U(s) \\ &= [1 - G(s)]U(s) \end{aligned} \tag{7-4-16}$$

The brute-force way to determine the steady-state error associated with the input $u(t)$ is to take the inverse Laplace transform of $E(s)$ in Eq. (7-4-16) and then let t approach infinity. However, there is a more elegant approach based on the final value theorem of the Laplace transform, as can be seen from the following result (Ogata, 1970):

Proposition 7-4-2: Steady-State Error. Let $e(\infty)$ denote the steady-state tracking error for the feedback system in Fig. 7-13, with $G_2(s) = 1$. If the poles of $sE(s)$ all lie in the open left half of the complex plane, then:

$$e(\infty) = \lim_{s \to 0} \{sE(s)\}$$

Thus there is no need to solve for $e(t)$ and then let t approach infinity. Instead, we can work directly with the expression for $E(s)$ in Eq. (7-4-16). The following example illustrates this technique:

Example 7-4-3: Steady-State Error

Consider the feedback system in Fig. 7-13, with $G_1(s) = k/[s(Ts + 1)]$ and $G_2(s) = 1$. Suppose $u(t) = 1(t)$, where $1(t)$ denotes the unit step function. Then, from Table 7-1,

$U(s) = 1/s$ and the steady-state tracking error for a unit step is:

$$\begin{aligned}
e(\infty) &= \lim_{s\to 0} \{sE(s)\} \\
&= \lim_{s\to 0} \{s[1 - G(s)]U(s)\} \\
&= \lim_{s\to 0} \left\{\frac{s[1 - G(s)]}{s}\right\} \\
&= \lim_{s\to 0} \{1 - G(s)\} \\
&= \lim_{s\to 0} \left\{1 - \frac{G_1(s)}{1 + G_1(s)}\right\} \\
&= \lim_{s\to 0} \left\{\frac{1}{1 + G_1(s)}\right\} \\
&= \lim_{s\to 0} \left\{\frac{1}{1 + k/[s(Ts + 1)]}\right\} \\
&= \lim_{s\to 0} \left\{\frac{s(Ts + 1)}{s(Ts + 1) + k}\right\} \\
&= 0
\end{aligned}$$

Thus when $G_1(s) = k/[s(Ts + 1)]$ and $G_2(s) = 1$, the feedback system in Fig. 7-13 has perfect steady-state tracking of a step input. Next, consider a more difficult input to track or follow, the unit ramp input $u(t) = t\,1(t)$. In this case, from Table 7-1 we have $U(s) = 1/s^2$ and the steady-state tracking error for the unit ramp is:

$$\begin{aligned}
e(\infty) &= \lim_{s\to 0} \{sE(s)\} \\
&= \lim_{s\to 0} \{s[1 - G(s)]U(s)\} \\
&= \lim_{s\to 0} \left\{\frac{s[1 - G(s)]}{s^2}\right\} \\
&= \lim_{s\to 0} \left\{\frac{1 - G(s)}{s}\right\} \\
&= \lim_{s\to 0} \left\{\frac{1 - G_1(s)/[1 + G_1(s)]}{s}\right\} \\
&= \lim_{s\to 0} \left\{\frac{1}{s[1 + G_1(s)]}\right\} \\
&= \lim_{s\to 0} \left\{\frac{1}{s(1 + k/[s(Ts + 1)])}\right\} \\
&= \lim_{s\to 0} \left\{\frac{(Ts + 1)}{s(Ts + 1) + k}\right\} \\
&= \frac{1}{k}
\end{aligned}$$

Thus the tracking of a unit ramp input is not perfect, but instead has a steady-state error of $1/k$. Of course, this error can be made small if the gain k can be increased.

7-4-3 Transient Performance

Steady-state tracking error is just one measure of performance for a feedback control system. Another important question is: How fast, and in what manner, does the error approach its steady-state value? In other words, what is the *transient performance* of the system? Roughly speaking, transient performance is a measure of the *speed* of the system, whereas steady-state tracking performance is a measure of the *accuracy*. To formulate criteria for transient performance, we examine the following second-order transfer function:

$$G(s) = \frac{k_0 \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{7-4-17}$$

Here the constants ω_n and ζ are referred to as the *undamped natural frequency* and the *damping ratio*, respectively. Since it does not affect transient performance, we can assume for convenience that the *DC gain* of the system, $G(0) = k_0$, is unity. The constants ζ and ω_n are design parameters that determine how fast the system responds to abrupt changes in the input. From Eq. (7-4-17), we see that the poles of $G(s)$ are at:

$$s_{1,2} = [-\zeta \pm (\zeta^2 - 1)^{1/2}]\omega_n \tag{7-4-18}$$

Applying Prop. 7-4-1, it follows that this second-order linear system is stable if and only if $\zeta\omega_n > 0$. Suppose $\omega_n > 0$ and $0 < \zeta < 1$, and consider the case of a unit step input, $u(t) = 1(t)$. Using Table 7-1, the Laplace transform of the output is $Y(s) = G(s)U(s) = G(s)/s$. By performing a partial fraction expansion of $Y(s)$ and using Table 7-1, one can obtain the following expression for $y(t)$ in this case:

$$y(t) = 1 - \frac{\exp(-\zeta\omega_n t)\sin[\beta\omega_n t + \text{atan2}(\beta, \zeta)]}{\beta} \tag{7-4-19a}$$

where

$$\beta \triangleq (1 - \zeta^2)^{1/2} \qquad 0 < \zeta < 1 \tag{7-4-19b}$$

Thus the steady-state part of the response to a unit step input is $y(t) = 1$, which implies zero steady-state tracking error. The transient part of the response is a damped sinusoidal oscillation with an amplitude of $1/\beta$, a frequency of $\beta\omega_n$, and a time constant of $1/(\zeta\omega_n)$, as shown in Fig. 7-14.

There are a number of parameters that can be used to measure the speed and sensitivity of this response. The time T_d that it takes the output to reach one-half of the steady-state value is called the *delay time*. The delay time T_d can be estimated from ω_n and ζ as follows (Kuo, 1982):

$$T_d \approx \frac{1 + 0.6\zeta + 0.15\zeta^2}{\omega_n} \qquad 0 < \zeta < 1 \tag{7-4-20}$$

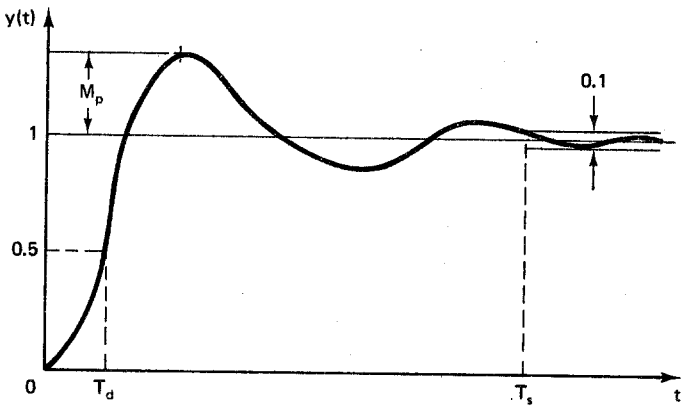


Figure 7-14 Transient response of underdamped linear system.

The delay time is a measure of the speed of the system. Another measure of speed is the time T_s that it takes for the output to settle to within ±5 percent of the steady-state value. This is referred to as the *settling time* and can be bounded as follows (Ogata, 1970):

$$T_s < \frac{4.7}{\zeta \omega_n} \qquad 0 < \zeta < 1 \tag{7-4-21}$$

From expressions (7-4-20) and (7-4-21) we see that the speed of the response can be increased by increasing the undamped natural frequency ω_n. A third parameter that is used to measure transient performance is the *peak overshoot* M_p of the steady-state value. The maximum overshoot is:

$$M_p = \exp\left(\frac{-\zeta\pi}{\beta}\right) \qquad 0 < \zeta < 1 \tag{7-4-22}$$

Recalling that $\beta = (1 - \zeta^2)^{1/2}$, we see that the maximum overshoot varies from $M_p = 0$ when $\zeta = 1$ to $M_p = 1$ when $\zeta = 0$. Generally, the maximum overshoot decreases as the damping ratio ζ increases. However, as ζ increases, the delay time also increases, as can be seen from Eq. (7-4-20). For many control systems, a damping ratio value of perhaps 0.7 or 0.8 is selected to limit the maximum overshoot yet maintain reasonable speed. However, for a robotic arm, it is important that *no overshoot* occur. This is because the tool must often be brought into contact with the environment. Here *any* overshoot of the destination position will cause a collision which could disturb the environment or damage the robot. To ensure that no overshoot occurs, we must choose $\zeta \geq 1$. The case $\zeta = 1$ is called *critical damping*, while $\zeta < 1$ is the *underdamped* case and $\zeta > 1$ is the *overdamped* case. Critical damping generates the fastest response that does not overshoot the steady-state position. When $\zeta = 1$, the response of the system to a unit step input simplifies to:

$$y(t) = 1 + (1 + \omega_n t)\exp(-\omega_n t) \qquad \zeta = 1 \tag{7-4-23}$$

From Eq. (7-4-18) we see that for critical damping, the transfer function has a double pole at $s = -\omega_n$. Since the exponential factor in Eq. (7-4-23) dominates the

factor $\omega_n t$, the speed of the system improves as the double pole at $s = -\omega_n$ is moved deeper into the left half plane.

An n-axis robotic arm can be modeled, very crudely, as n separate second-order linear systems whose parameters vary with time. For example, the moment of inertia about each joint varies as the configuration of the arm changes. This is particularly true of the proximal joints near the base of the arm. A linear controller can be used to ensure that, in the worst case, the damping ratio of the closed-loop control system will always be at least unity. This "tuning" of the parameters of the control system is done to keep the arm from overshooting. However, since the design is based on the worst case, there will be configurations of the arm for which the system is highly overdamped, and in these cases the response of the arm will be quite sluggish.

Exercise 7-4-1: Damping Ratio. Consider the following second-order linear system, where $y(t)$ denotes an angular position variable. Suppose the moment of inertia D varies over the range $1 \leq D \leq 10$. What is the minimum value of b that will ensure that this system is never underdamped? Use Eq. (7-4-20) to estimate the range of delay times in this case.

$$Dy^{(2)}(t) + by^{(1)}(t) + y(t) = u(t)$$

7-5 SINGLE-AXIS PID CONTROL

A common approach to robot control that is in use in many commercial robots is single-axis PID control. To describe PID control mathematically, let $r(t) \in \mathbf{R}^n$ denote the desired *reference input* that we would like the joint variables to follow, and let $e(t)$ represent the *tracking error* at time t. That is:

$$e(t) \triangleq r(t) - q(t) \tag{7-5-1}$$

Ideally, $e(t)$ is identically zero, but in practice it varies from zero, particularly when the reference input $r(t)$ changes rapidly. A common technique for controlling a robotic manipulator is to employ n independent controllers, one for each axis, as follows:

$$\tau(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \dot{e}(t) \tag{7-5-2}$$

Here $\{K_P, K_I, K_D\}$ are $n \times n$ diagonal matrices, which means that each axis is controlled *separately*. Note that the PID control signal τ consists of terms Proportional to the error, the Integral of the error, and the Derivative of the error; hence the name. The diagonal matrices $\{K_P, K_I, K_D\}$ are design parameters called the *proportional* gain, *integral* gain, and *derivative* gain, respectively. Refer to Fig. 7-15 for a block diagram representation of a single-axis PID controller. Here s denotes the differentiation operation, and $1/s$ denotes the integration operation. When $n = 1$, the diagonal gain matrices reduce to 1×1 scalar gains $\{k_P, k_I, k_D\}$.

The expression for τ in Eq. (7-5-2) is called a *control law*. It produces a torque *set point* τ that serves as input to the torque regulator subsystem in Fig. 7-15. The

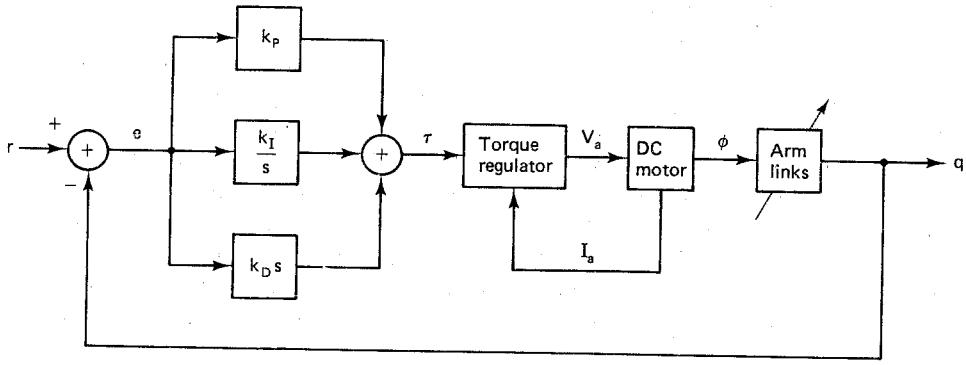


Figure 7-15 Single-axis PID controller.

torque regulator controls the motor torque by sensing the armature winding current I_a and varying the armature winding voltage V_a. The load for the kth PID controller includes link k as well as the distal links all the way out to the end of the arm. Therefore the load on each joint changes rapidly during normal operation of the robot. This variation is indicated symbolically in Fig. 7-15 with a diagonal arrow through the arm links.

7-5-1 DC Motor and Load

To analyze single-axis PID control, we must model the subsystems in Fig. 7-15 in more detail. First consider the DC motor and load. Although some robots (for example, the Microbot Alpha II and Intelledex 660) use stepper motors for actuators, most commercial electric-drive robots use DC servomotors. A schematic diagram of an armature-controlled DC servomotor with mechanical load is shown in Fig. 7-16.

To develop a dynamic model of this actuator, we start with the armature winding circuit on the left-hand side of Fig. 7-16. Applying Kirchhoff's voltage law around the armature winding yields:

$$L_a \dot{I}_a(t) + R_a I_a(t) + V_{\text{emf}}(t) = V_a(t) \tag{7-5-3}$$

Here the input $V_a(t)$ is the *applied armature* voltage, while $I_a(t)$ is the armature

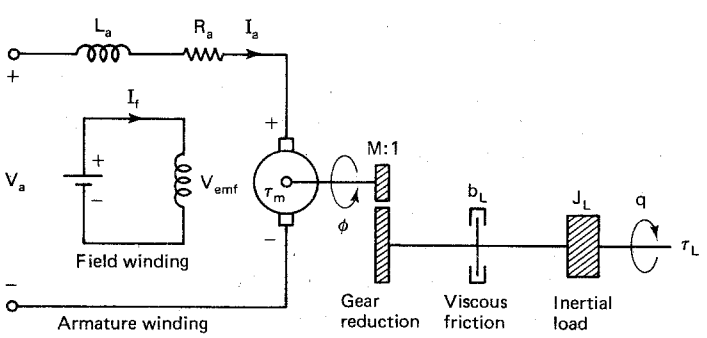


Figure 7-16 Armature-controlled DC motor and load.

winding current. The parameters R_a and L_a are the *resistance* and *inductance* of the armature winding, respectively. Finally, $V_{\text{emf}}(t)$ is called the *back emf* voltage. The back emf (electromotive force) voltage is an internal voltage that counteracts $V_a(t)$ and builds up as the motor shaft speed increases:

$$V_{\text{emf}}(t) = k_b \dot{\phi}(t) \tag{7-5-4}$$

The constant $k_b > 0$ is called the *back emf constant*.

Next we examine the mechanical part of the system in Fig. 7-16. Applying Newton's second law, the equation of motion for the rotating mass is:

$$J\ddot{\phi}(t) + b\dot{\phi}(t) = \tau_m(t) \tag{7-5-5}$$

Here $\phi(t)$ denotes angular position of the motor shaft and $\tau_m(t)$ denotes the torque developed at the shaft. The parameters J and b are the effective *moment of inertia* and *viscous friction coefficient*, respectively, referred to the motor shaft. If J_m and b_m represent the moment of inertia and viscous friction coefficient for the motor and J_L and b_L denote corresponding quantities for the load, then:

$$J = J_m + M^2 J_L \tag{7-5-6}$$

$$b = b_m + M^2 b_L \tag{7-5-7}$$

$$q = M\phi \tag{7-5-8}$$

Here M is the *gear reduction ratio* between the high-speed motor shaft ϕ and the load shaft q.

The electrical and mechanical subsystems are coupled to one another through an algebraic torque equation. In general, the torque developed at the motor shaft is proportional to the product of two currents, the armature winding current I_a and the field winding current I_f. However, in an armature-controlled DC motor, the field winding current is constant. Consequently, the torque developed at the motor shaft is:

$$\tau_m(t) = k_a I_a(t) \tag{7-5-9}$$

The constant $k_a > 0$ is called the *torque constant* of the motor. To develop a transfer function representation of the DC motor we take Laplace transforms of Eqs. (7-5-3), (7-5-4), and (7-5-9), assuming zero initial conditions, and combine the results into one equation, which yields:

$$\frac{(L_a s + R_a)\tau_m(s)}{k_a} + sk_b\phi(s) = V_a(s) \tag{7-5-10}$$

Next we take Laplace transforms of both sides of Eq. (7-5-5), again assuming zero initial conditions. This results in the following equation for the mechanical part of the system in Fig. 7-16:

$$(Js^2 + bs)\phi(s) = \tau_m(s) \tag{7-5-11}$$

Since the DC motor and load are driven by a torque regulator system in Fig. 7-15, it is useful to develop a *torque transfer function*, $G_L(s) \triangleq Q(s)/\tau_L(s)$, which relates the load shaft position to the torque developed at the load shaft. Assuming that

there is no backlash or elastic deformation in the gears, the work done by the load shaft equals the work done by the motor shaft, $\tau_L q = \tau_m \phi$. Thus, from Eq. (7-5-8), the load shaft torque is:

$$\tau_L = \frac{\tau_m}{M} \tag{7-5-12}$$

Recall that $M = \phi/q$ is the gear reduction ratio from the motor shaft to the load shaft. If the desired load shaft torque is regarded as an input signal, then from Eqs. (7-5-8), (7-5-11), and (7-5-12), we see that the torque transfer function is:

$$G_L(s) = \frac{M^2}{s(Js + b)} \tag{7-5-13}$$

Thus the transfer function relating the input $\tau_L(s)$ to the output $Q(s)$ is a second-order system with a pole at the origin. From Eq. (7-5-13) we see that the *time constant* of the torque transfer function is:

$$T_m = \frac{J}{b} \tag{7-5-14}$$

Recall that J is the effective moment of inertia and b is the effective coefficient of viscous friction. If we ignore the effects of viscous friction by setting $b = 0$, the torque transfer function reduces to a pure inertia, $G_L(s) = M^2/(Js^2)$.

A block diagram of the torque transfer function for the DC motor and load is shown in Fig. 7-17.

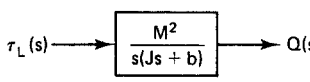


Figure 7-17 Torque transfer function of DC motor and load.

Exercise 7-5-1: Voltage Transfer Function. Suppose the armature voltage V_a is taken as the input. Use Eqs. (7-5-8), (7-5-10), and (7-5-11) to find an expression for the armature voltage transfer function $G_V(s) \triangleq Q(s)/V_a(s)$.

7-5-2 Torque Regulator

Next consider the torque regulator subsystem in Fig. 7-15. The torque of a DC motor can be controlled indirectly by varying the applied armature voltage V_a. Recall from Eq. (7-5-9) that the torque is proportional to the armature winding current I_a. To regulate the torque (or armature current), we insert a small *current-sensing resistor* R_s in series with the armature winding in order to generate a feedback voltage V_s as shown in Fig. 7-18.

The key control element in the feedback system in Fig. 7-18 is the oscillator, OSC. The output of the oscillator, V_{PWM}, is a pulse train with *pulse width* w and *period* T, where T is selected to be small in comparison with the motor time constant T_m. Thus V_{PWM} is a *pulse-width-modulated* (PWM) signal, as shown in Fig. 7-19.

The signal V_{PWM} switches drive transistors Q_1 and Q_2 in Fig. 7-18 on and off. Depending upon the sign of τ, one of the transistors will be turned off while the

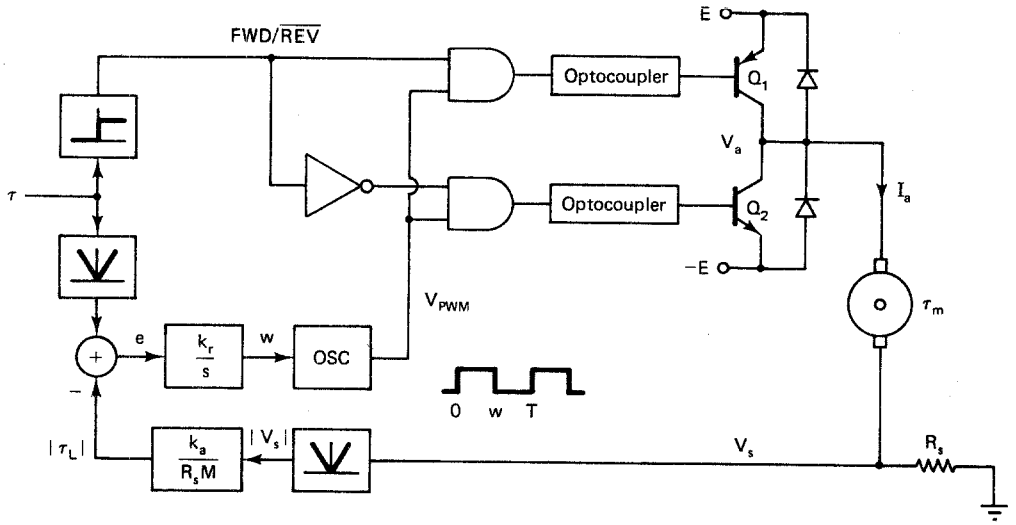


Figure 7-18 Torque regulator system.

other is pulsed on and off by V_{PWM}. Thus the sign of τ controls the direction in which the motor shaft turns, while the pulse width w controls the torque developed at the shaft. Torque is controlled by changing the average value of the armature voltage $V_a(t)$. Given a DC supply voltage of E, the *average value* of the magnitude of the mature voltage is $\bar{V}_a = wE/T$, where $0 \leq w \leq T$. Taking Laplace transforms, assuming zero initial conditions, this yields:

$$\bar{V}_a(s) = \frac{W(s)E}{T} \tag{7-5-15}$$

It is the average armature voltage, or DC component of $V_a(t)$, that is important, because a DC motor is essentially a low-pass filter that removes the fundamental and higher harmonics of the periodic signal $V_a(t)$ as long as the PWM period T is small in comparison with the motor time constant T_m in Eq. (7-5-14).

Note from Fig. 7-18 that the torque regulator employs *integral* feedback control with a gain of k_r. To develop the closed-loop transfer function for the torque regulator, we start at $W(s)$ and work backward. Referring to Fig. 7-18, and using Eqs. (7-5-9) and (7-5-12), we have:

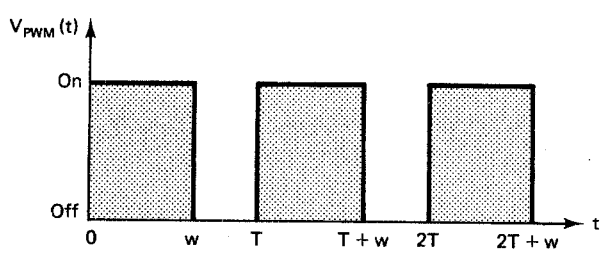


Figure 7-19 Pulse-width-modulated (PWM) control signal.

$$
\begin{aligned}
W(s) &= \frac{k_r E(s)}{s} \\
&= k_r \frac{|\tau(s)| - k_a |V_s(s)|/(MR_s)}{s} \\
&= k_r \frac{|\tau(s)| - |k_a V_s(s)/(MR_s)|}{s} \\
&= k_r \frac{|\tau(s)| - |k_a I_a(s)/M|}{s} \\
&= k_r \frac{|\tau(s)| - |\tau_m(s)/M|}{s} \\
&= k_r \frac{|\tau(s)| - |\tau_L(s)|}{s}
\end{aligned} \tag{7-5-16}
$$

We can now substitute Eq. (7-5-16) in Eq. (7-5-15), and we can replace $\bar{V}_a(s)$ with $V_a(s)$, assuming that $T \ll T_m$. The magnitude signs can be dropped because the sign of V_a is controlled directly by τ through the switching circuit in Fig. 7-18. Thus the (approximate) relationship between the load torque and the applied armature voltage is:

$$
V_a(s) \approx \frac{k_r[\tau(s) - \tau_L(s)]E}{Ts} \qquad T \ll T_m \tag{7-5-17}
$$

We see that applied armature voltage is proportional to the integral of the torque error $\tau - \tau_L$, with the constant of proportionality being $k_r E/T$. We can now combine Eqs. (7-5-10), (7-5-11), and (7-5-17) and solve for the torque regulator transfer function $G_\tau(s) \triangleq \tau_L(s)/\tau(s)$, the result being:

$$
G_\tau(s) \approx \frac{k_a k_r E(Js + b)}{Ts(Js + b)(L_a s + R_a) + k_a k_b Ts + k_a k_r E(Js + b)} \tag{7-5-18}
$$

Hence the torque regulator is a third-order system with a DC gain of unity. It is often the case that the armature winding inductance L_a is sufficiently small that it can be neglected. Setting $L_a = 0$ in Eq. (7-5-18) and redefining the coefficients, we get the following simplified transfer function for the torque regulator:

$$
G_\tau(s) \approx \frac{k_1(Js + b)}{s^2 + k_2 s + k_1 b} \qquad L_a \approx 0 \tag{7-5-19}
$$

Here the parameters $k_1 > 0$ and $k_2 > 0$ can be computed from the original physical parameters of the torque regulator and motor as follows:

$$
k_1 \triangleq \frac{k_a k_r E}{R_a JT} \tag{7-5-20}
$$

$$
k_2 \triangleq \frac{TR_a b + Tk_a k_b + k_a k_r EJ}{R_a JT} \tag{7-5-21}
$$

Note from Eq. (7-5-18) that the DC gain of the closed-loop system ($s = 0$) is unity. Thus, if $\tau(t)$ is constant, then in the steady state $\tau_L = \tau$, which means that the torque delivered at the load shaft q equals the requested torque τ. This is based on the assumption that $0 \leq w \leq T$, since saturation occurs when the pulse width exceeds the period. Saturation of the PWM signal arises when the magnitude of the requested torque exceeds the maximum torque $\tau_{\max}$ that can be delivered to the load shaft. A DC motor produces maximum torque when the motor stalls. In this case both ϕ and I_a are constant. From Eqs. (7-5-3) and (7-5-4), we see that the armature current during a stall condition is $I_{\max} = E/R_a$, where E is the DC supply voltage. Thus, from Eqs. (7-5-9) and (7-5-12), the maximum load shaft torque is:

$$\tau_{\max} = \frac{k_a E}{R_a M} \tag{7-5-22}$$

The requested torque must satisfy $|\tau(t)| \leq \tau_{\max}$ to avoid saturation of the PWM torque regulator circuit. In terms of the reference trajectory $r(t)$, this typically means that the speed of the robot along the desired path may have to be reduced, because otherwise more torque may be required than the actuator is capable of delivering.

Recall from Prop. 7-4-1 that the torque regulator is stable if and only if the poles of $G_\tau(s)$ lie in the open left half of the complex plane. The following exercise shows that this will be the case as long as the torque regulator gain k_r is positive.

Exercise 7-5-2: Torque Regulator Stability. Show that the simplified transfer function of the torque regulator in Eq. (7-5-19) is stable if and only if $k_r > 0$.

The torque regulator gain k_r is a design parameter than can be selected by the user. The transfer function of the torque regulator can be simplified further if k_r is made large. Indeed, from Eqs. (7-5-19) through (7-5-21), we have:

$$\begin{aligned}
\lim_{k_r \to \infty} G_\tau(s) &= \lim_{k_r \to \infty} \left[\frac{k_1(Js + b)}{s^2 + k_2 s + k_1 b} \right] \\
&= \lim_{k_r \to \infty} \left[\frac{k_1(Js + b)}{s^2 + Jk_1 s + k_1 b} \right] \\
&= \lim_{k_r \to \infty} \left[\frac{Js + b}{s^2/k_1 + Js + b} \right] \\
&= 1
\end{aligned} \tag{7-5-23}$$

Thus for large values of the gain k_r, the torque regulator can be modeled approximately by its steady-state characteristic $\tau_L = \tau$, assuming $|\tau| \leq \tau_{\max}$. This is equivalent to assuming that the dynamics of the torque regulator are fast in comparison with the rest of the system in Fig. 7-15. In this case we can model the torque regulator using the following algebraic equation, which takes saturation into account:

$$\tau_L \approx \tau_{\max} \text{ sat} \left(\frac{\tau}{\tau_{\max}} \right) \tag{7-5-24}$$

Here sat denotes the *unit saturation* function, defined as sat $(a) = a$ if $|a| \leq 1$, sat $(a) = 1$ if $a > 1$, and sat $(a) = -1$ if $a < -1$. Thus the input/output model of the torque regulator is as shown in Fig. 7-20, where the break-point τ_{max} depends on the torque constant, the DC supply voltage, the armature winding resistance, and the gear ratio as indicated in Eq. (7-5-22).

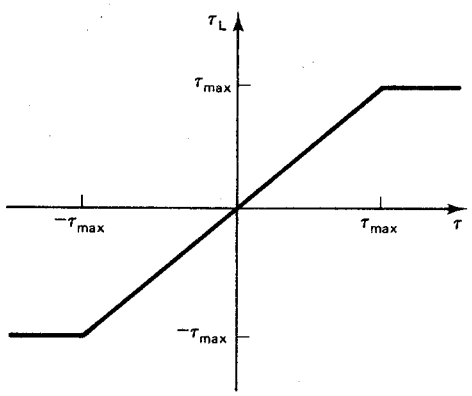


Figure 7-20 Input/output model of torque regulator.

To simplify the subsequent analysis, we will assume that k_r is sufficiently large that the torque regulator can be modeled by its steady-state characteristic in Fig. 7-20. In addition, we will often assume that τ_{max} is sufficiently large that it can be ignored.

7-5-3 PID Transfer Function

We can now combine the simplified model of the torque regulator in Fig. 7-20 with the torque transfer function of the DC motor and load in Fig. 7-17. Assuming τ_{max} is sufficiently large that it can be ignored for the reference trajectory of interest, the result is a simplified linear model of the single-axis PID controller as shown in Fig. 7-21.

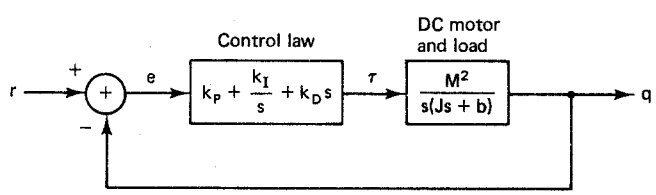


Figure 7-21 Transfer function model of single-axis PID controller.

The transfer function model in Fig. 7-21 is based on a number of assumptions, including the assumption that the load, embodied in J and b, does not change. Consequently, the transfer function model is strictly applicable only to a single-axis robot and, moreover, that axis should be aligned with the gravitational field, because we have not included in the model disturbance torques due to gravity.

Proposition 7-5-1: PID Transfer Function. The closed-loop transfer func-

tion $G_{\text{PID}}(s) \triangleq Q(s)/R(s)$ of the single-axis PID controller in Fig. 7-21 is:

$$G_{\text{PID}}(s) = \frac{M^2(k_D s^2 + k_P s + k_I)}{Js^3 + (M^2 k_D + b)s^2 + M^2 k_P s + M^2 k_I}$$

Proof. First we find the open-loop transfer function $G_1(s)$ of the single-axis PID controller. From Fig. 7-21, we have:

$$\begin{aligned} G_1(s) &= \frac{(k_P + k_I/s + k_D s)M^2}{s(Js + b)} \\ &= \frac{M^2(k_D s^2 + k_P s + k_I)}{s^2(Js + b)} \\ &\triangleq \frac{b(s)}{a(s)} \end{aligned}$$

Note that the single-axis PID controller is a unity-gain negative feedback system where $G_2(s) = 1$. Thus, from Eq. (7-4-15), the closed-loop transfer function is:

$$\begin{aligned} G_{\text{PID}}(s) &= \frac{G_1(s)}{1 + G_1(s)} \\ &= \frac{b(s)/a(s)}{1 + [b(s)/a(s)]} \\ &= \frac{b(s)}{a(s) + b(s)} \\ &= \frac{M^2(k_D s^2 + k_P s + k_I)}{s^2(Js + b) + M^2(k_D s^2 + k_P s + k_I)} \\ &= \frac{M^2(k_D s^2 + k_P s + k_I)}{Js^3 + (M^2 k_D + b)s^2 + M^2 k_P s + M^2 k_I} \end{aligned}$$

Hence, the single-axis PID controller can be modeled as a third-order linear system. Note from the expression in Prop. 7-5-1 that the closed-loop DC gain ($s = 0$) is unity. Consequently, the steady-state tracking error for a step input, $r(t) = 1(t)$, is zero, assuming the closed-loop system is stable. From Prop. 7-4-1, the closed-loop transfer function $G_{\text{PID}}(s)$ is stable if and only if the poles of $G_{\text{PID}}(s)$ all lie in the open left half of the complex plane. Since the denominator of $G_{\text{PID}}(s)$ is a third-degree polynomial, we cannot factor it directly. Instead, we use an indirect method called the *Routh-Hurwitz test* (Ogata, 1970), which leads to the following stability criterion:

Proposition 7-5-2: PID Stability. The PID controller in Fig. 7-21 is stable if the controller gains satisfy the following inequalities: $k_I \geq 0$, $k_D > -b/M^2$, and

$$(M^2 k_D + b)k_P > Jk_I$$

Proof. From Prop. 7-4-1, $G_{\text{PID}}(s)$ is stable if and only if the poles of $G_{\text{PID}}(s)$ all

lie in the open left half of the complex plane. Poles of $G_{PID}(s)$ are zeros of the denominator polynomial that are not canceled by zeros of the numerator polynomial. From Prop. 7-4-1, the denominator polynomial is:

$$\begin{aligned} d(s) &= Js^3 + (M^2k_D + b)s^2 + M^2k_Ps + M^2k_I \\ &\triangleq a_0s^3 + a_1s^2 + a_2s + a_3 \end{aligned}$$

To determine constraints on the controller gains which guarantee stability, we form the Routh array (Ogata, 1970) from the coefficients of $d(s)$ as follows:

$$\begin{array}{ll} a_0 & a_2 \\ a_1 & a_3 \\ b_1 & 0 \\ c_1 & \end{array}$$

Here the last two rows of the Routh array are computed from the first two rows as follows (Ogata, 1970):

$$\begin{aligned} b_1 &= \frac{a_1a_2 - a_0a_3}{a_1} \\ &= \frac{(M^2k_D + b)M^2k_P - JM^2k_I}{M^2k_D + b} \\ &= \frac{M^2((M^2k_D + b)k_P - Jk_I)}{M^2k_D + b} \\ c_1 &= \frac{b_1a_3 - a_10}{b_1} \\ &= a_3 \\ &= M^2k_I \end{aligned}$$

A necessary and sufficient condition for the zeros of $d(s)$ to have negative real parts is that all entries in the first column of the Routh array be nonzero and of the same sign. Since $a_0 = J$, it follows that the remaining entries in the first column must all be positive. Now $a_1 = M^2k_D + b$. Thus the constraint $a_1 > 0$ is satisfied if $k_D > -b/M^2$. It then follows that the constraint $b_1 > 0$ is satisfied if:

$$(M^2k_D + b)k_P > Jk_I$$

The last constraint, $c_1 > 0$, is satisfied if $k_I > 0$. The inequality $k_I > 0$ can be relaxed to $k_I \geq 0$. Indeed, if $k_I = 0$, then $d(s)$ has a zero at $s = 0$. However when $k_I = 0$, the numerator of $G_{PID}(s)$ also has a zero at $s = 0$ which cancels the zero of $d(s)$. Thus it is sufficient that $k_I \geq 0$.

It is of interest to note from Prop. 7-5-2 that we can set $k_I = 0$ and/or $k_D = 0$ and still maintain stability. Thus PI, PD, and P controllers are all potentially useful special cases of the PID controller in Fig. 7-21.

Exercise 7-5-3: PD Control. Suppose the integral gain is $k_I = 0$, which corresponds to a PD controller. Using Prop. 7-5-1, find the simplified transfer function $G_{PD}(s)$.

Exercise 7-5-4: Steady-State Error. Use Prop. 7-4-2 to find the steady-state tracking error of the PID controller for the following inputs, assuming the controller gains satisfy Prop. 7-5-2 with $k_I > 0$:

1. Step: $r(t) = 1(t)$
2. Ramp: $r(t) = t\,1(t)$
3. Parabola: $r(t) = t^2 1(t)$

Exercise 7-5-5: Transfer Function. Suppose the torque regulator is modeled using Eq. (7-5-19) rather than $G_\tau(s) = 1$. Find the resulting transfer function $G_{PID}(s)$. Show that $G_{PID}(s)$ reduces to the expression in Prop. 7-5-1 in the limit as $k_r \to \infty$.

Exercise 7-5-6: Stability. Use the generalized transfer function from Exercise 7-5-5 to find constraints on the controller gains which guarantee stability. Show that your results reduce to those of Prop. 7-5-2 in the limit as $k_r \to \infty$.

One advantage of the PID controller is that it is very simple to implement. Each axis can have its own separate PID control loop. Here the proportional gain term initiates a correction in the torque set-point signal τ whenever the error e is nonzero. The derivative gain term serves to increase the margin of stability. Finally, the integral gain term tends to reduce the size of the steady-state tracking error, because any residual error will grow when it is integrated and therefore lead to a correction in τ.

Example 7-5-1: Three-Axis SCARA Robot

Consider the three-axis SCARA robot shown in Fig. 7-4. From Eqs. (7-5-1) and (7-5-2), the following is an implementation of a single-axis PID controller:

$$e = r - q$$

$$\tau_1(t) = k_P^1 e_1(t) + k_I^1 \int_0^t e_1(\tau)\,d\tau + k_D^1 \dot{e}_1(t)$$

$$\tau_2(t) = k_P^2 e_2(t) + k_I^2 \int_0^t e_2(\tau)\,d\tau + k_D^2 \dot{e}_2(t)$$

$$\tau_3(t) = k_P^3 e_3(t) + k_I^3 \int_0^t e_3(\tau)\,d\tau + k_D^3 \dot{e}_3(t)$$

The controller gains $\{k_P^1, k_I^1, k_D^1\}$, $\{k_P^2, k_I^2, k_D^2\}$, and $\{k_P^3, k_I^3, k_D^3\}$ can be chosen independently for the three axes using Prop. 7-5-2 as a general guideline.

The main drawback of the PID controller in robotics is that the *load* seen by the motor or actuator of each joint can vary rapidly and substantially. This is particu-

larly true of the proximal joints near the base, where the moments of inertia and the loading due to gravity can change by an order of magnitude. As the control task becomes more demanding, involving high-speed movement of large payloads, the performance of n independent PID controllers begins to deteriorate because of a lack of knowledge about the dynamics of the manipulator. The analysis of the PID controller presented here assumes that the load does not change. Consequently, it is strictly applicable only to a one-axis robot, and moreover, that axis should be aligned with the field of gravity so that there is no gravitational disturbance. Thus the PID controller analysis is applicable, for example, to the vertical base axis of a multiaxis robot if all the other joints are locked so that the arm behaves as one rigid link. Alternatively, if the load on a joint varies, this can be modeled as a disturbance, where the disturbances are compensated for (to some extent) by feedback.

7-6 PD-GRAVITY CONTROL

One of the deficiencies of single-axis PID control is that it does not account for the effects of gravity. In this section we examine an n-axis PD-type controller that includes an explicit nonlinear gravity compensation term in the control law. Recall from Prop. 7-2-1 that the state equations of a robotic manipulator are of the following form:

$$\dot{q} = v \tag{7-6-1a}$$

$$\dot{v} = D^{-1}(q)[\tau - c(q, v) - h(q) - b(v)] \tag{7-6-1b}$$

We can control the joint variables indirectly by using the state feedback formulation shown in Fig. 7-22. The equation $\tau = f(q, v, r)$ is called a state feedback control law. Our goal is to find a control law f such that the joint position $q(t)$ of the closed-loop system closely tracks the reference input $r(t)$.

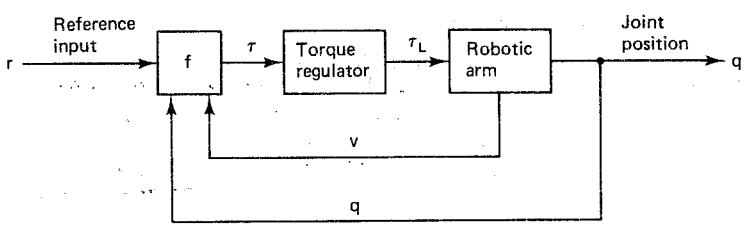


Figure 7-22 State feedback control of a robotic arm.

The simplest and perhaps most important special case occurs when the reference input $r(t) = a$ for some constant set point $a \in \mathbf{R}^n$. This is called the *regulator* problem, or set-point control problem. Here we must find a control law which will drive the system from an arbitrary initial condition to the steady-state solution $q(t) = a$. In addition, we want the controller to regulate the arm about the set-point solution in spite of disturbances that may occur, such as variations in the load at the end of the arm. As a candidate for a feedback control law, consider the following proportional-plus-derivative (PD) formulation with gravity compensation:

$$e = r - q \tag{7-6-2a}$$

$$\tau = Ke + L\dot{e} + h(q) \tag{7-6-2b}$$

Here K and L are symmetric $n \times n$ positive-definite matrices called the *position gain* matrix and the *velocity gain* matrix, respectively. A matrix K is *positive-definite* if and only if $z^TKz > 0$ for $z \neq 0$. We see that τ consists of terms proportional to the error and the derivative of the error, plus a nonlinear gravity term, $h(q)$. If we substitute Eq. (7-6-2) in Eq. (7-6-1), this yields the following closed-loop equations of motion for the system in Fig. 7-22:

$$\dot{q} = v \tag{7-6-3a}$$

$$\dot{v} = D^{-1}(q)[K(r - q) + L(\dot{r} - v) - c(q, v) - b(v)] \tag{7-6-3b}$$

Let us suppose $r(t) = a$ for $t \geq 0$ for some set point a. To analyze the resulting equilibrium points of the closed-loop system, we set the right-hand side of Eq. (7-6-3) to zero and solve for q and v. This yields $v = 0$ and $D^{-1}(q)[K(a - q) - c(q, 0) - b(0)] = 0$. But from Eqs. (6-4-8) and (6-3-3), $c(q, 0) = 0$ and $b(0) = 0$. Thus $K(a - q) = 0$, where K is positive-definite. Consequently, the closed-loop system in Eq. (7-6-3) has a single equilibrium point at:

$$\hat{x}^T = [a^T, 0^T] \tag{7-6-4}$$

Now $x(t) = \hat{x}$ is the desired steady-state solution of the arm when the reference input is $r(t) = a$. Therefore, to show that the PD-gravity controller has the desired closed-loop behavior, it is sufficient to show that the equilibrium point $\hat{x}$ is asymptotically stable and that its domain of attraction encompasses the entire state space.

Proposition 7-6-1: PD-Gravity Control. Let $x^T = [q^T, v^T]$ be the solution of the robotic arm in Eq. (7-6-1), assuming that τ is computed using the PD-gravity control law in Eq. (7-6-2). If $r(t) = a$ for $t \geq 0$, then the equilibrium point $\hat{x}^T = [a^T, 0^T]$ is asymptotically stable and the domain of attraction is $\Omega = \mathbf{R}^{2n}$. Thus, for each $x(0) \in \mathbf{R}^{2n}$:

$$x(t) \rightarrow \hat{x} \qquad \text{as} \qquad t \rightarrow \infty$$

Proof. First we perform a change of variable to move the equilibrium point to the origin. Let $z \triangleq q - a$. Then, if $r(t) = a$ for $t \geq 0$, the closed-loop equations in Eq. (7-6-3) can be recast in terms of z and v as:

$$\dot{z} = v \tag{7-6-5a}$$

$$\dot{v} = -D^{-1}(z + a)[Kz + Lv + c(z + a, v) + b(v)] \tag{7-6-5b}$$

Since $c(z + a, 0) = 0$ and $b(0) = 0$, it follows that $(z, v) = (0, 0)$ is an equilibrium point of the transformed system. To show that this equilibrium point is asymptotically stable, we use Liapunov's second method. Recall from Eq. (6-2-19) that $v^TD(q)v/2$ represents the kinetic energy of the arm. As a candidate Liapunov function for the closed-loop system, consider:

$$V_L(z, v) = \frac{z^TKz + v^TD(z + a)v}{2} \tag{7-6-6}$$

The manipulator inertia tensor $D(q)$ is a continuously differentiable positive-definite matrix. Since the position gain K is also a positive-definite matrix, it follows from Def. 7-3-5 that $V_L(z, v)$ is a valid Liapunov function. To apply Prop. 7-3-3, we must show that $\dot{V}_L(z(t), v(t))$ decreases along solutions of the state equation. Since K is symmetric, evaluating $\dot{V}_L(z, v)$ along the solution of Eq. (7-6-5) yields:

$$\begin{aligned}
\dot{V}_L(z, v) &= \left(\frac{d}{dt}\right)\frac{z^T K z + v^T D(z + a)v}{2} \\
&= z^T K \dot{z} + v^T D(z + a)\dot{v} + \frac{v^T \dot{D}(z + a)v}{2} \\
&= z^T K v - v^T[Kz + Lv + c(z + a, v) + b(v)] + \frac{v^T \dot{D}(z + a)v}{2} \\
&= (z^T K v)^T - v^T[Kz + Lv + c(z + a, v) + b(v)] + \frac{v^T \dot{D}(z + a)v}{2} \\
&= v^T K^T z - v^T[Kz + Lv + c(z + a, v) + b(v)] + \frac{v^T \dot{D}(z + a)v}{2} \\
&= v^T K z - v^T[Kz + Lv + c(z + a, v) + b(v)] + \frac{v^T \dot{D}(z + a)v}{2} \\
&= \frac{v^T \dot{D}(z + a)v}{2} - v^T[Lv + c(z + a, v) + b(v)] \\
&= v^T\left[\frac{\dot{D}(z + a)v}{2} - c(z + a, v)\right] - v^T[Lv + b(v)] \qquad (7\text{-}6\text{-}7)
\end{aligned}$$

The Coriolis and centrifugal force term $c(q, v)$ can be expressed in terms of $\dot{D}(q)$ and a matrix $F(q, v)$ where $F(q, v)$ is skew-symmetric (Koditschek, 1984):

$$c(z + a, v) = \frac{[\dot{D}(z + a, v) - F(z + a, v)]v}{2} \qquad (7\text{-}6\text{-}8)$$

Substituting Eq. (7-6-8) in Eq. (7-6-7) yields the following expression for $\dot{V}_L(z, v)$ along the solution of Eq. (7-6-5):

$$\dot{V}_L(z, v) = \frac{v^T F(z + a, v)v}{2} - v^T[Lv + b(v)] \qquad (7\text{-}6\text{-}9)$$

But since $F(q, v)$ is skew-symmetric, we have $F^T(q, v) = -F(q, v)$. Therefore the term containing $F(q, v)$ does not contribute to $\dot{V}_L(q, v)$, because:

$$\begin{aligned}
v^T F(z + q, v)v &= \frac{v^T F(z + a, v)v}{2} + \frac{v^T F(z + a, v)v}{2} \\
&= \frac{v^T F(z + a, v)v}{2} + \frac{[v^T F(z + a, v)v]^T}{2}
\end{aligned}$$

$$= \frac{v^T F(z + a, v)v}{2} + \frac{v^T F^T(z + a, v)v}{2}$$

$$= \frac{v^T F(z + a, v)v}{2} - \frac{v^T F(z + a, v)v}{2}$$

$$= 0 \tag{7-6-10}$$

Combining Eqs. (7-6-9) and (7-6-10), and recalling Eq. (6-3-3), the expression for $\dot{V}_L(z, v)$ along the solution of the closed-loop state equation in Eq. (7-6-5) reduces to:

$$\begin{aligned}
\dot{V}_L(z, v) &= -v^T[Lv + b(v)] \\
&= -[v^T Lv + v^T b(v)] \\
&= -\left[v^T Lv + \sum_{k=1}^{n} v_k b_k(v)\right] \\
&= -\left\{v^T Lv + \sum_{k=1}^{n} v_k\left[b_k^v v_k + \text{sgn}\,(v_k)[b_k^d + (b_k^s - b_k^d)\exp(-|v_k|/\epsilon)]\right]\right\} \\
&= -\left\{v^T Lv + \sum_{k=1}^{n}\left[b_k^v (v_k)^2 + |v_k|[b_k^d + (b_k^s - b_k^d)\exp(-|v_k|/\epsilon)]\right]\right\}
\end{aligned} \tag{7-6-11}$$

Recall that L is positive-definite, and the friction coefficients are all nonnegative. Consequently, $\dot{V}_L(z + a, v) \leq 0$ along solutions of Eq. (7-6-5). Thus condition 1 of Prop. 7-3-3 is satisfied. To establish condition 2 of Prop. 7-3-3, note from Eq. (7-6-11) that, since K and L are positive-definite:

$$\begin{aligned}
\dot{V}_L(z(t) + a, v(t)) \equiv 0 \quad &\Rightarrow \quad v(t) \equiv 0 \\
&\Rightarrow \quad \dot{v}(t) \equiv 0 \\
&\Rightarrow \quad D^{-1}(z(t) + a, 0)Kz(t) \equiv 0 \\
&\Rightarrow \quad z(t) \equiv 0
\end{aligned} \tag{7-6-12}$$

Thus condition 2 is satisfied, and from Prop. 7-3-3, the equilibrium point $(z, v) = (0, 0)$ is asymptotically stable. The domain of attraction is the set Ω_ρ where:

$$\begin{aligned}
\Omega_\rho &= \{(z, v): V_L(z, v) < \rho\} \\
&= \left\{(z, v): \frac{z^T Kz + v^T D(z + a)v}{2} < \rho\right\}
\end{aligned} \tag{7-6-13}$$

But $V_L(z, v)$ is a Liapunov function on Ω_ρ and conditions 1 and 2 of Prop. 7-3-3 are satisfied on Ω_ρ for every $\rho > 0$. Furthermore, since K and $D(z + a, v)$ are positive-definite matrices, $V_L(z, v) \to \infty$ as $\|z\| + \|v\| \to \infty$. Thus the domain of attraction is the entire state space $\Omega = \mathbf{R}^{2n}$. That is, the equilibrium point $(q, v) = (a, 0)$ is asymptotically stable in the large.

Proposition 7-6-1 is a rather remarkable result. It says that once the effects of gravity have been removed, the nonlinear robotic arm can be successfully controlled using a simple linear PD controller. The only constraint for the closed-loop system to be stable is that the controller gains K and L be symmetric positive-definite matrices. Refer to Fig. 7-23 for a block diagram of the PD-gravity controller for a robotic manipulator.

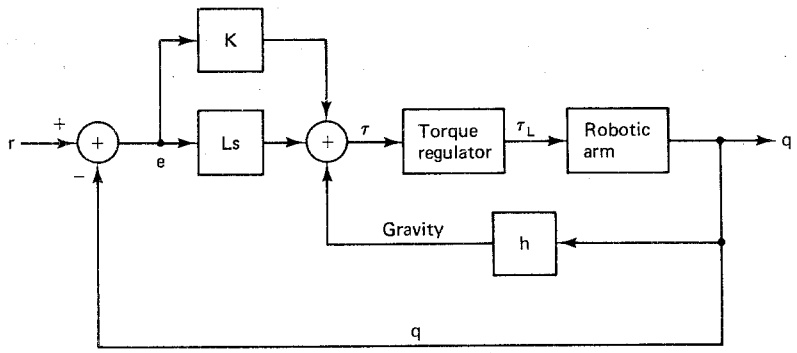


Figure 7-23 PD-gravity controller.

The practical significance of the PD-gravity control law in Eq. (7-6-2) lies in the fact that it requires no detailed knowledge of the manipulator inertia tensor $D(q)$, the Coriolis and centrifugal coupling vector $c(q, v)$, or the friction vector $b(v)$. It does require knowledge of the gravity loading vector $h(q)$, but this is relatively easy to determine, at least in comparison. In an actual implementation of Eq. (7-6-2), the exact value of $h(q)$ will not be known. Instead:

$$h(q) = \tilde{h}(q) + \Delta h(q) \tag{7-6-14}$$

where the term $\tilde{h}(q)$ represents an *estimate* of $h(q)$ and $\Delta h(q)$ represents the *error* in the estimate caused by uncertainty in the exact mass and geometry of the links. Using $\tilde{h}(q)$ in place of $h(q)$ in the control law will cause the performance of the controller to deteriorate somewhat. For example, there may be a small variation in the location of the equilibrium point, which means that the steady-state tracking error in response to a step input will no longer be zero.

It is of interest to compare the nonlinear n-axis PD-gravity controller in Fig. 7-23 with the linear single-axis PID controller in Fig. 7-21. In a sense, Prop. 7-6-1 can be regarded as a theoretical justification of the linear PID control method. To see this, suppose we choose the positive-definite gain matrices in Eq. (7-6-2) to be diagonal matrices with positive diagonal elements:

$$K = \text{diag}\,\{k_1, k_2, \ldots, k_n\} \tag{7-6-15a}$$

$$L = \text{diag}\,\{l_1, l_2, \ldots, l_n\} \tag{7-6-15b}$$

If we consider an axis for which $h_i(q) = 0$, then the PD-gravity control law is identical to the single-axis PID control law when $k_I = 0$. Setting $k_I = 0$ in Prop. 7-5-2, we see that the constraints on the controller gains reduce to $k_P > 0$ and $k_D > -b/J$. From Prop. 7-6-1, the corresponding constraints are $k_i > 0$ and $l_i > 0$.

Thus Prop. 7-6-1 (a sufficient condition) puts somewhat more conservative bounds on the controller gains. In the event that the friction b is negligible, the two sets of constraints become identical.

Example 7-6-1: Two-Axis Planar Articulated Robot

As an example of a PD-gravity controller, consider the two-axis planar articulated arm shown in Fig. 7-3. Suppose K and L are diagonal, as in Eq. (7-6-15). Using Eq. (7-6-2) and the expression for $h(q)$ developed in Eq. (7-3-4), the following is an implementation of a PD-gravity controller in this case:

$$e = r - q$$

$$\tau_1 = k_1 e_1 + l_1 \dot{e}_1 + g_0 \left[\left(\frac{m_1}{2} + m_2 \right) a_1 C_1 + \frac{m_2 a_2 C_{12}}{2} \right]$$

$$\tau_2 = k_2 e_2 + l_2 \dot{e}_2 + \frac{g_0 m_2 a_2 C_{12}}{2}$$

More generally, K and L can be any real symmetric matrices with positive eigenvalues. By varying the values of the components of K and L, we can control the relative speeds with which the two joints are driven to the desired set point. Of course, if the gains are too large, then the magnitude of the requested torques will exceed the maximum torque τ_{max} that can be delivered by the joint actuators.

Example 7-6-2: Three-Axis SCARA Robot

As a second example of a PD-gravity controller, consider the three-axis SCARA robot shown in Fig. 7-4. Again, suppose K and L are diagonal, as in Eq. (7-6-15). Because of the geometry of the SCARA robot, there is no gravity loading on the revolute joints. From Eq. (7-2-10), we see that the gravity loading on the prismatic joint is simply $h_3(q) = m_3 g_0$. Thus, from Eq. (7-6-2), the following is an implementation of a PD-gravity controller in this case:

$$e = r - q$$

$$\tau_1 = k_1 e_1 + l_1 \dot{e}_1$$

$$\tau_2 = k_2 e_2 + l_2 \dot{e}_2$$

$$\tau_3 = k_3 e_2 + l_3 \dot{e}_2 + m_3 g_0$$

This controller was simulated on a computer using parameter values roughly similar to the Adept One robot (Wu, 1988). In particular, the joint mass vector and joint link length vector were:

$$m = [8.0, 5.0, 3.0]^T \text{ kg}$$

$$a = [0.8, 0.3, 0.3]^T \text{ m}$$

In addition, a load mass of $m_L = 1.5$ kg was added to the mass of link 3 as an "unknown" disturbance. The coefficients of friction used for the simulation were:

$$b_k^v = 1.0 \qquad 1 \le k \le 3$$

$$b_k^d = 0.0 \qquad 1 \le k \le 3$$

$$b_k^s = 15.0 \qquad 1 \le k \le 3$$

$$\epsilon = 0.1$$

The reference trajectory for the tool tip was a circle in the plane $x = 0.1$ with a radius of 0.3 centered at (0.1, 0.2, 0.5). The motion along this circular path started with a constant acceleration for 0.5 sec, followed by zero acceleration (constant speed) for 0.5 sec, followed by constant deceleration for 0.5 sec to close the circle. Thus the total travel time for the tool-tip trajectory $p(t)$ was $T = 1.5$ sec. The inverse kinematic equations from Chap. 3 were used to determine a corresponding joint-space reference trajectory $r(t)$. This trajectory was sampled every 2 msec to produce a sequence of 750 set points. From Prop. 7-6-1, we see that any positive values can be used for the PD controller gains. The values used for the simulation were:

$$k_i = 10{,}000 \qquad 1 \leq k \leq 3$$

$$l_i = 200 \qquad 1 \leq k \leq 3$$

For a denominator polynomial of $d_i(s) = s^2 + l_i s + k_i$, this corresponds to critical damping, $\zeta = 1$, with an undamped natural frequency of $\omega_n = 100$.

The results of the simulation are shown in Fig. 7-24. Note that the controller performed quite well in this case in spite of the presence of the disturbance m_L, which gave rise to the following error in the estimate of the gravity loading term:

$$\Delta h(q) = [0, 0, m_L g_0]^T$$

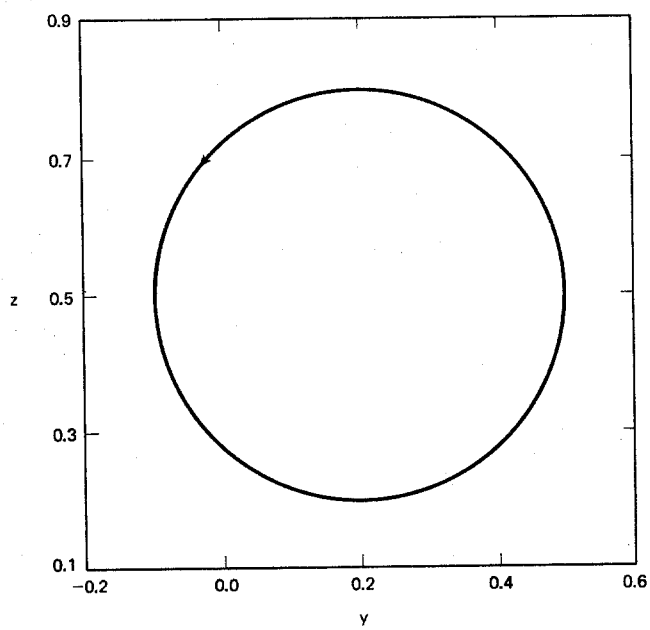


Figure 7-24 PD-plus-gravity control of three-axis SCARA robot.

The maximum tracking error between the tool-tip position and the corresponding point on the reference circle was 1.17 mm. Most of the error (1.13 mm) occurred in the z coordinate of the tool tip, which is controlled by joint 3. The control torques needed to drive the SCARA robot along the circular path are shown in Fig. 7-25.

Note that there is some oscillation at the start of the trajectory during the constant acceleration phase. The control torques for the first two joints are somewhat smoother than that for joint 3, where most of the tracking error occurs.

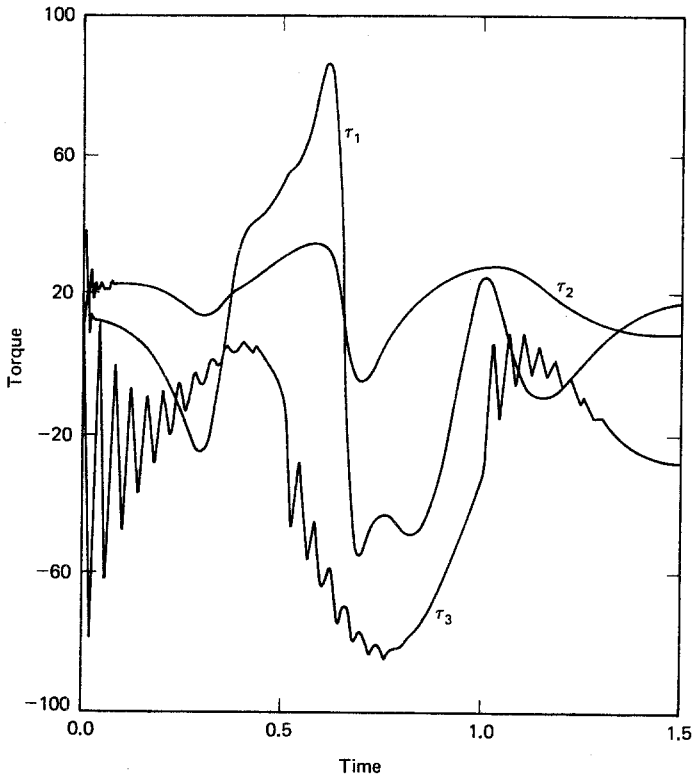


Figure 7-25 Control torques applied to three-axis SCARA robot.

7-7 COMPUTED-TORQUE CONTROL

Recall that the set point, or regulator, control problem corresponds to the case when the reference input $r(t)$ is constant. We now examine the more general case when $r(t)$ is variable. This is called the *servo* control problem. We assume that the planned trajectory $r(t)$ is sufficiently smooth that it has at least two derivatives. Bejczy (1974) proposed a technique for solving the robot servo problem called the *computed-torque* method. It is an approach that makes direct use of the complete dynamic model of the manipulator to cancel not only the effects of gravity, but also Coriolis and centrifugal force, friction, and the manipulator inertia tensor.

A realistic implementation of computed-torque control must rely on *estimates* of the robotic arm parameters rather than the exact values. Let $\{\tilde{D}, \tilde{c}, \tilde{h}, \tilde{b}\}$ denote the estimated values of the manipulator inertia tensor, Coriolis and centrifugal force vector, gravity loading vector, and frictional force vector, respectively. That is,

$$D = \tilde{D} + \Delta D \tag{7-7-1a}$$

$$c = \tilde{c} + \Delta c \tag{7-7-1b}$$

$$h = \tilde{h} + \Delta h \tag{7-7-1c}$$

$$b = \tilde{b} + \Delta b \tag{7-7-1d}$$

The terms $\{\Delta D, \Delta c, \Delta h, \Delta b\}$ denote the *errors* in the estimates of the robotic arm parameters. These errors will always be present to some degree. To attempt to cancel the nonlinear terms of the dynamic model of the manipulator, the following state feedback control law can be used:

$$\tau = \tilde{h}(q) + \tilde{c}(q, v) + \tilde{b}(v) + \tilde{D}(q)[K(r - q) + L(\dot{r} - v) + \ddot{r}] \tag{7-7-2}$$

Here K and L are again symmetric $n \times n$ positive-definite gain matrices associated with the position feedback and velocity feedback terms, respectively. Notice that estimates of all of the terms of the original open-loop dynamic model of the arm appear in the feedback control law. The linear terms inside the square brackets represent the desired dynamics that we would like the arm to have. To examine the effects of this control law, we formulate the closed-loop equations of the arm in terms of the error vector $e = r - q$. This leads to the following result:

Proposition 7-7-1: Computed-Torque Control. Let $x^T = [q^T, v^T]$ be the solution of the robotic arm in Eq. (7-6-1), assuming that τ is computed using the computed-torque control law in Eq. (7-7-2). Then the error $e = r - q$ is a solution of the following second-order system:

$$\ddot{e} + (I - D^{-1}\,\Delta D)(L\dot{e} + Ke) = D^{-1}(\Delta h + \Delta c + \Delta b + \Delta D\ddot{r})$$

Proof. Starting with the open-loop state equations of the arm in Eq. (7-6-1) and using the computed-torque control law in Eq. (7-7-5), we have:

$$\begin{aligned}
\ddot{e} &= \ddot{r} - \ddot{q} \\
&= \ddot{r} - \dot{v} \\
&= \ddot{r} - D^{-1}(\tau - h - c - b) \\
&= \ddot{r} + D^{-1}\{\Delta h + \Delta c + \Delta b - \tilde{D}[K(r - q) + L(\dot{r} - v) + \ddot{r}]\} \\
&= \ddot{r} + D^{-1}[\Delta h + \Delta c + \Delta b - \tilde{D}(K\tilde{e} + L\dot{e} + \ddot{r})] \\
&= \ddot{r} + D^{-1}(\Delta h + \Delta c + \Delta b) - D^{-1}\tilde{D}(Ke + L\dot{e} + \ddot{r}) \\
&= \ddot{r} + D^{-1}(\Delta h + \Delta c + \Delta b) - (I - D^{-1}\Delta D)(Ke + L\dot{e} + \ddot{r}) \\
&= -(I - D^{-1}\Delta D)(Ke + L\dot{e}) + D^{-1}(\Delta h + \Delta c + \Delta b + \Delta D\ddot{r})
\end{aligned}$$

Refer to Fig. 7-26 for a block diagram of a computed-torque controller for a robotic arm. The right half of the diagram attempts to cancel the undesired nonlinear dynamics, while the left half inserts the desired linear dynamics.

It is of interest to observe what happens when the estimates of the robotic arm parameters are exact. In this case we see that the error equation in Prop. 7-7-1 reduces to a *linear* second-order equation that is *independent* of the robotic arm parameters. Thus the nonlinear terms are completely canceled in this idealized case.

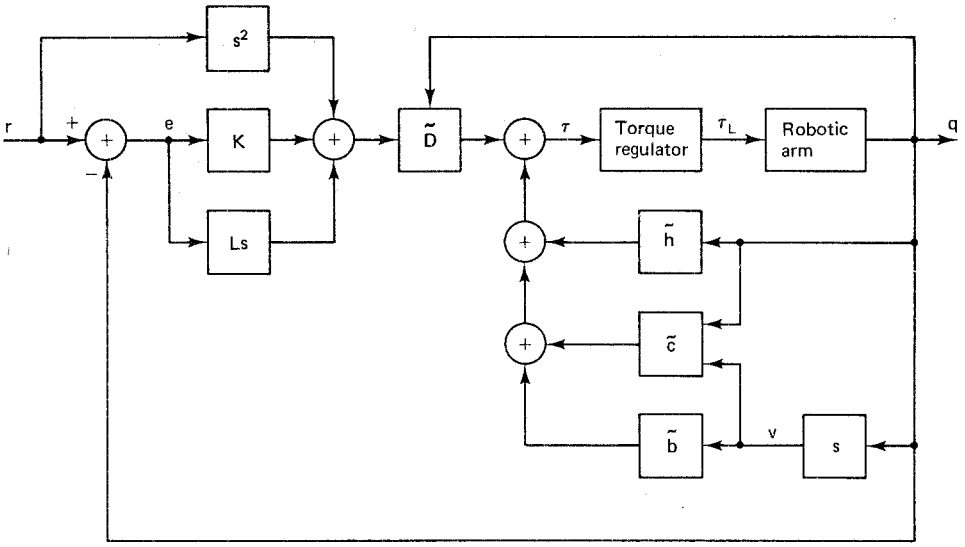


Figure 7-26 Computed-torque controller.

If the gain matrices K and L are diagonal, then the closed-loop equations of motion are not only linear, but they are also *uncoupled* from one another.

Exercise 7-7-1: Critical Damping. Suppose the gain matrices K and L are diagonal as in Eq. (7-6-15) and the estimates of the robotic arm parameters are exact. Find an expression for the roots of the characteristic polynomial of the error equation for the ith axis. For what values of k_i and l_i is the system critically damped?

If reasonable servo performance is to be obtained, the control signal $\tau(t)$ should be updated at a rate of at least 100 to 200 Hz. Thus there are at most 5 to 10 msec available to evaluate the control law in Eq. (7-7-2). However, computing values for the robotic arm parameters $\{D, c, h, b\}$ can require many operations. Consequently, for a real-time implementation of a computed-torque controller, an efficient implementation of the recursive Newton-Euler equations should be used (Luh et al., 1980).

There are a number of practical drawbacks to the computed-torque control method. The first, of course, is that our estimates of the robotic arm parameters $\{D, c, h, b\}$ are never exact. The estimates often contain significant error because many of the individual parameters cannot be measured directly; instead, they must be inferred from other measurements. Another deficiency which prevents complete cancellation of nonlinear effects is the observation that the model itself is never complete. For example, it does not take into account such things as flexibility in the links, gear backlash, or the load at the end of the arm generated by the object being manipulated. A numerical implementation of the control law will also contain round off-error due to finite-word-length arithmetic. This error can be reduced by using higher-precision arithmetic, but at the expense of increased computational time. Fi-

nally, if the reference input $r(t)$ varies too rapidly, it will generate terms in the computed torque $\tau(t)$ whose magnitudes exceed the limits of the joint actuators thereby causing saturation of the torque regulator.

Example 7-7-1: Two-Axis Planar Articulated Robot

As an example of a computed-torque controller, consider the two-axis planar articulated arm shown in Fig. 7-3. Suppose K and L are diagonal as in Eq. (7-6-15). Using Eq. (7-7-2) and the dynamic model in Eq. (7-2-4), the following is an implementation of a computed-torque controller in this case:

$$e = r - q$$

$$\tau_1 = g_0\left[\left(\frac{m_1}{2} + m_2\right)a_1C_1 + \frac{m_2a_2C_{12}}{2}\right] - m_2a_1a_2S_2\left(\dot{q}_1\dot{q}_2 + \frac{\dot{q}_2^2}{2}\right) + b_1(\dot{q}_1)$$
$$+ \left[\left(\frac{m_1}{3} + m_2\right)a_1^2 + m_2a_1a_2C_2 + \frac{m_2a_2^2}{3}\right](k_1e_1 + l_1\dot{e}_1 + \ddot{r}_1)$$
$$+ \left(\frac{m_2a_1a_2C_2}{2} + \frac{m_2a_2^2}{3}\right)(k_2e_2 + l_2\dot{e}_2 + \ddot{r}_2)$$

$$\tau_2 = \frac{g_0m_2a_2C_{12}}{2} + \frac{m_2a_1a_2S_2\dot{q}_1^2}{2} + b_2(\dot{q}_2)$$
$$+ \left(\frac{m_2a_1a_2C_2}{2} + \frac{m_2a_2^2}{3}\right)(k_1e_1 + l_1\dot{e}_1 + \ddot{r}_1) + \left(\frac{m_2a_2^2}{3}\right)(k_2e_2 + l_2\dot{e}_2 + \ddot{r}_2)$$

Again, by varying the values of k_i and l_i, we can control the relative speed of the controller as long as the gains are not so large that the magnitude of the requested torque exceeds the maximum torque that can be delivered by the joint actuators. Comparing the computed-torque controller with the PD-gravity controller in Example 7-6-1, we see that there is a substantial increase in complexity even for this simple case when $n = 2$.

Example 7-6-2: Three-Axis SCARA Robot

As a second example of a computed-torque controller, consider the three-axis SCARA robot shown in Fig. 7-4. Again, suppose K and L are diagonal as in Eq. (7-6-15). Using Eq. (7-7-2) and the dynamic model in Eq. (7-2-10), the following is an implementation of a computed-torque controller in this case:

$$e = r - q$$

$$\tau_1 = -a_1a_2S_2\left[(m_2 + 2m_3)\dot{q}_1\dot{q}_2 - \left(\frac{m_2}{2} + m_3\right)\dot{q}_2^2\right] + b_1(\dot{q}_1)$$
$$+ \left[\left(\frac{m_1}{3} + m_2 + m_3\right)a_1^2 + (m_2 + 2m_3)a_1a_2C_2\right.$$
$$\left. + \left(\frac{m_2}{3} + m_3\right)a_2^2\right](k_1e_1 + l_1\dot{e}_1 + \ddot{r}_1)$$
$$- \left[\left(\frac{m_2}{2} + m_3\right)a_1a_2C_2 + \left(\frac{m_2}{3} + m_3\right)a_2^2\right](k_2e_2 + l_2\dot{e}_2 + \ddot{r}_2)$$

$$\tau_2 = \left(\frac{m_2}{2} + m_3\right)a_1a_2S_2\dot{q}_1^2 + b_2(\dot{q}_2)$$
$$- \left[\left(\frac{m_2}{2} + m_3\right)a_1a_2C_2 + \left(\frac{m_2}{3} + m_3\right)a_2^2\right](k_1e_1 + l_1\dot{e}_1 + \ddot{r}_1)$$
$$+ \left(\frac{m_2}{3} + m_3\right)a_2^2(k_2e_2 + l_2\dot{e}_2 + \ddot{r}_2)$$

$$\tau_3 = -g_0m_3 + b_3(\dot{q}_3) + m_3(k_3e_3 + l_3\dot{e}_3 + \ddot{r}_3)$$

This controller was simulated on a computer using the same parameter values, load, and reference trajectory as described in Example 7-6-2 (Wu, 1988). The results of the simulation are shown in Fig. 7-27. Note that the controller performed quite well in this case in spite of the presence of the disturbance m_L, which caused the estimates of the robotic parameters to be inexact. The errors in the estimates due to the increased mass of link 3 were:

$$\Delta D(q) = m_L\begin{bmatrix} a_1^2 + 2a_1a_2C_2 + a_2^2 & -(a_1a_2C_2 + a_2^2) & 0 \\ -(a_1a_2C_2 + a_2^2) & a_2^2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Delta c(q, v) = m_L[a_1a_2S_2(v_2^2 - 2v_1v_2), a_1a_2S_2v_1^2, 0]^T$$

$$\Delta h(q) = m_L[0, 0, -g_0]^T$$

The maximum tracking error between the tool-tip position and the corresponding point on the reference circle in this case was 1.25 mm. Most of the error (1.24 mm) occurred in the y coordinate of the tool tip in this case. The computed torques used to drive the SCARA robot along the circular path are shown in Fig. 7-28. Note the high-frequency content in all three control torques in comparison with the smoother PD-gravity control torques in Fig. 7-25.

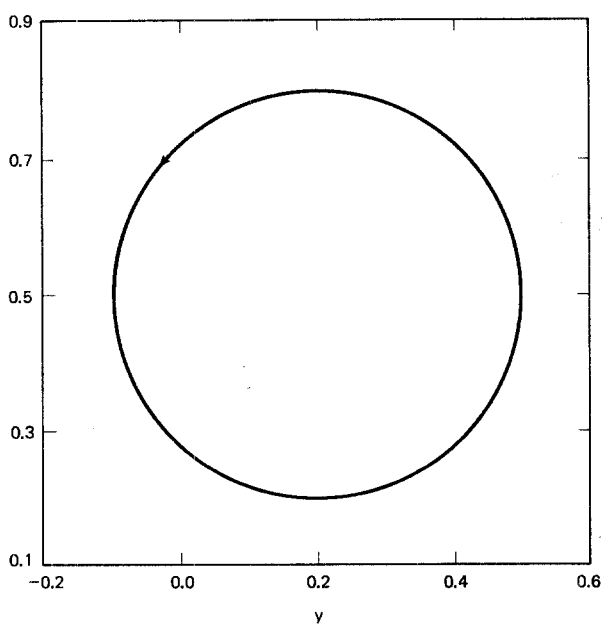


Figure 7-27 Computed-torque control of three-axis SCARA robot.

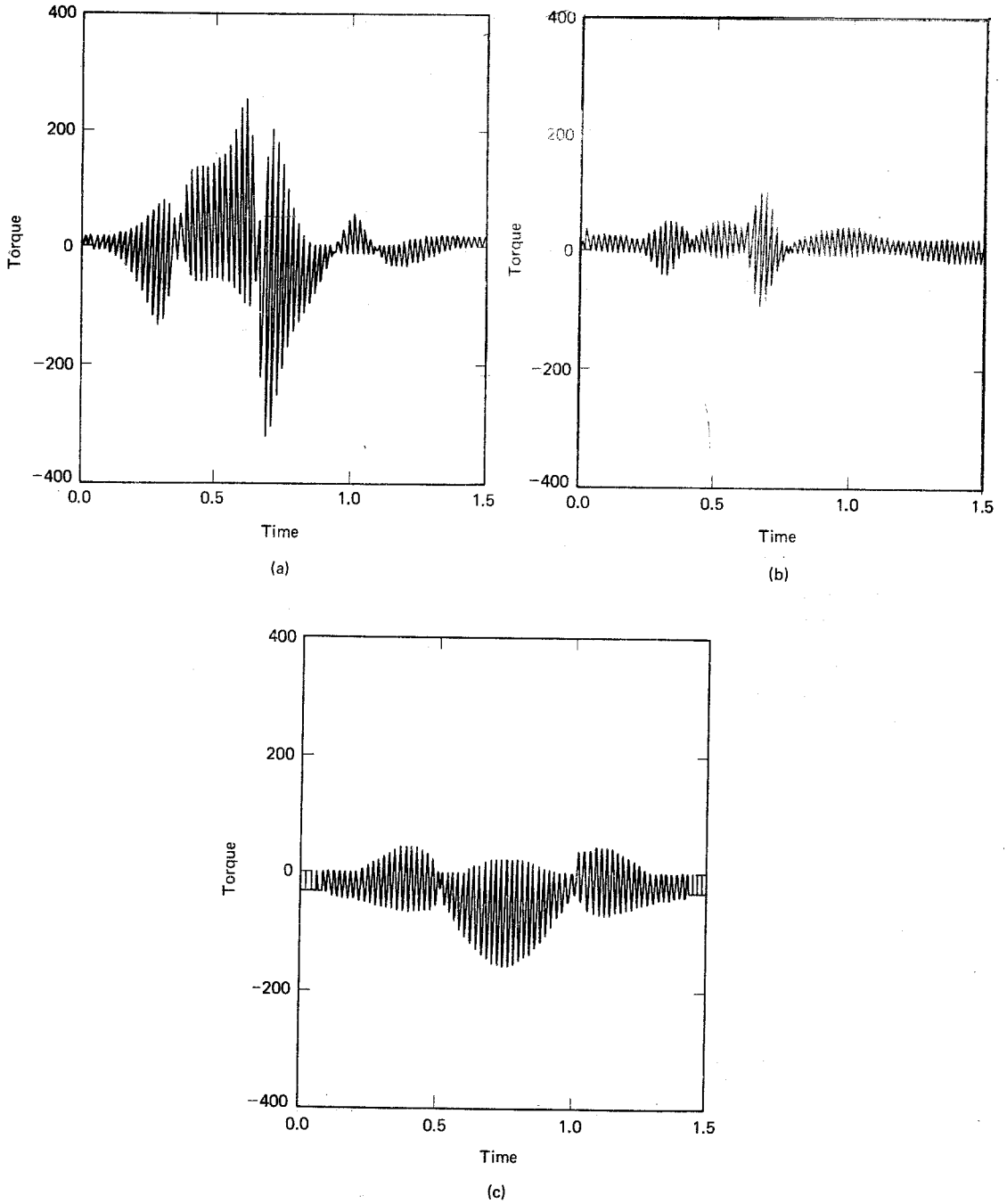


Figure 7-28 Control torques computed for three-axis SCARA robot (a) axis 1, (b) axis 2, (c) axis 3.

The simulation in Fig. 7-27 is based on the assumption that the actuators can deliver all of the torque requested in Fig. 7-28. To investigate what happens when the torque regulator saturates, consider the case when the limit in Fig. 7-20 is set to $\tau_{max} = 70$. The resulting response of the "saturated" computed-torque controller is shown in Fig. 7-29. Clearly, the performance deteriorates substantially if the computed torque gets "clipped" when it exceeds τ_{max}. In this case the maximum tool-tip tracking error is an unacceptable 88.9 mm.

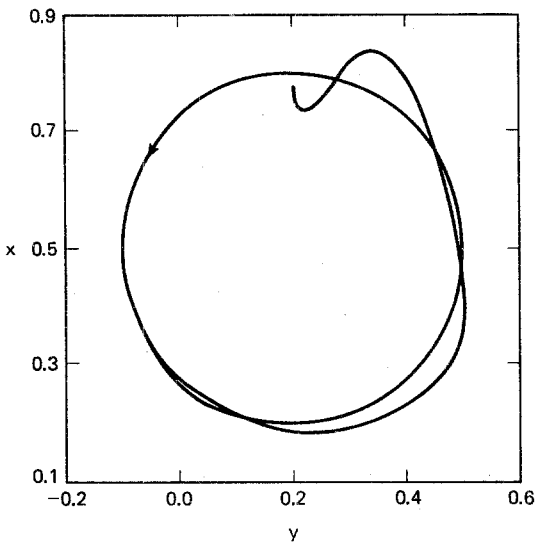


Figure 7-29 Computed-torque control with saturation.

7-8 VARIABLE-STRUCTURE CONTROL

The computed-torque control method relies on knowledge of the robotic arm parameters and therefore suffers from sensitivity to errors in the estimates of these parameters. A number of approaches have been proposed to develop controllers that are more *robust* so that their performance is not sensitive to modeling errors. For example, the use of *adaptive control* techniques has been examined. Here the control law is continuously updated to adapt to changes in the parameters of the system being controlled. Dubowski and DesForges (1979) proposed a model-reference-type adaptive controller for a robotic arm as shown in Fig. 7-30. Here a linear second-order reference model is used for each joint of the manipulator. The response of the manipulator is compared with the response of the reference model, and the error between the two is used to update the controller gains in an attempt to make the robot respond like the reference model. Koivo and Guo (1983) developed an adaptive controller for a robotic arm based on an autoregressive model that used a least-squares criterion to obtain the best fit to the manipulator input/output data. Lee and Chung (1984) linearized the manipulator equation about the planned robot trajectory, and then applied adaptive control techniques to the resulting linear time-varying system.

Adaptive control methods are based on the assumption that the parameters of

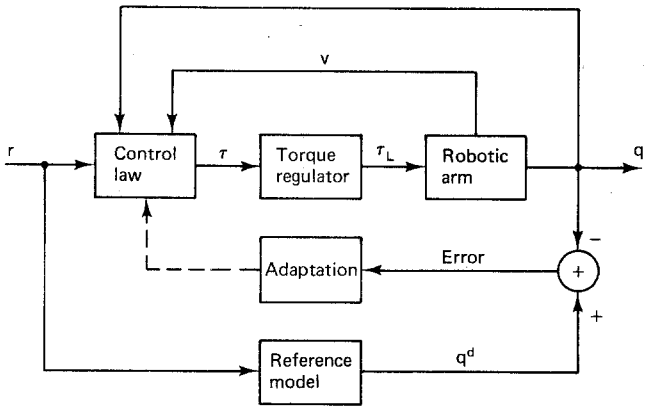


Figure 7-30 Model-reference adaptive control of a robotic arm.

the system being controlled do not change too rapidly in comparison with the system time constants. These techniques have proved quite effective when applied, for example, to chemical processes where the process parameters undergo gradual change. However, for robotic manipulators, the system parameters such as the inertia and the effects of gravity tend to change rapidly as the arm moves from one configuration to another. For this reason, the application of adaptive control methods to robotic manipulators has thus far enjoyed only limited success.

The basic idea behind adaptive control is that the controller gains gradually change as the *parameters* of the system being controlled evolve. It is also possible to change the control signal abruptly on the basis of the *state* of the system being controlled. Control systems of this type are referred to as *variable-structure* systems (Itkis, 1976). A block diagram of a variable-structure controller for a robotic arm is shown in Fig. 7-31.

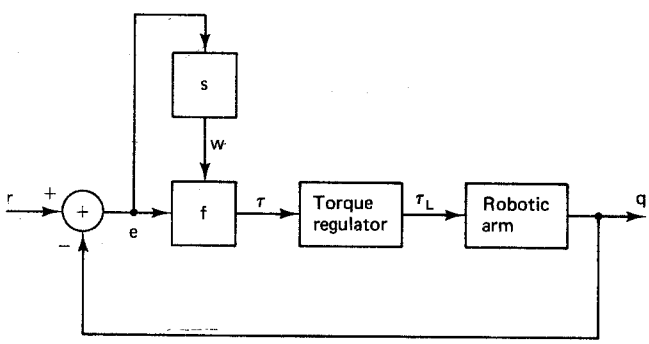


Figure 7-31 Variable-structure control of a robotic arm.

To apply variable-structure control, we do not have to know the exact robotic arm parameters, instead only *bounds* on these parameters. Variable-structure controllers are robust in the sense that they are insensitive to errors in the estimates of the parameters as long as reliable bounds on the parameters are known. To formulate a variable-structure control law, it is helpful to first recast the state equations in terms of the tracking error and its derivative. Suppose the reference input $r(t)$ is

sufficiently smooth that it has at least one derivative. Define the state vector as $x^T \triangleq [e^T, w^T]$, where $e = r - q$ and $w = \dot{e}$. Then, from Eq. (7-6-1), the closed-loop equations of motion of a robotic manipulator in terms of e and w are:

$$\dot{e} = w \tag{7-8-1a}$$

$$\dot{w} = \ddot{r} - D^{-1}(r - e)[\tau - h(r - e) - c(r - e, \dot{r} - w) - b(\dot{r} - w)] \tag{7-8-1b}$$

$$\tau = f(e, w) \tag{7-8-1c}$$

Given a reference trajectory $r(t)$, the objective is to find a variable-structure control law $\tau = f(x)$ such that the solution of the closed-loop system satisfies $x(t) \to 0$ as $t \to \infty$. Consider the following linear constraint on the state variables:

$$\sigma(x) \triangleq Fe + w = 0 \tag{7-8-2}$$

Here F can be any positive-definite matrix. For example, F might be a diagonal matrix with positive diagonal elements:

$$F = \text{diag}\{f_1, f_2, \ldots, f_n\} \tag{7-8-3}$$

The set of all x such that $\sigma(x) = 0$ is a $(2n - 1)$-dimensional subspace or "hyperplane" in $\mathbf{R}^{2n}$ which we refer to as the *switching surface*. The switching surface divides the state space into two regions. If $\sigma(x) > 0$, then we are on one side of the switching surface and the control law will have one form; if $\sigma(x) < 0$, then we are on the other side of the switching surface and the control law will have a different form. Thus the controller changes structure when the state of the system crosses the switching surface. The simplest special case of a switching surface arises when $n = 1$. In this case we see from Eqs. (7-8-2) and (7-8-3) that $\sigma(x) = 0$ corresponds to a line through the origin with a slope of $-f_1$, as shown in Fig. 7-32.

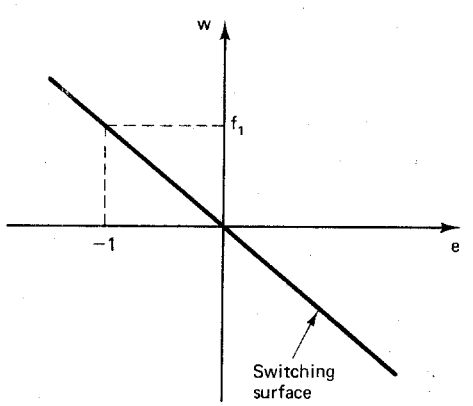


Figure 7-32 The switching surface when $n = 1$.

Our objective is to devise a control law $\tau = f(x)$ which will drive the system to the switching surface in a finite time and then constrain the system to stay on the switching surface. When the system is operating on the switching surface, we say that it is in the *sliding mode*. The dynamics of the system simplify substantially in the sliding mode. If $\sigma(x) = 0$, then from Eq. (7-8-2) the state equation for $\dot{e}$ reduces to:

$$\dot{e} + Fe = 0 \tag{7-8-4}$$

Thus when the system is in the sliding mode the tracking error is *independent* of the robotic arm parameters. The solution depends only on the matrix F, which is a design parameter called the *sliding-mode gain* matrix. Recall from Eq. (7-8-3) that F is a diagonal matrix with positive diagonal elements. Consequently, the sliding-mode equation is not only linear but also uncoupled, and the solution is:

$$e_k(t) = \exp(-f_k t)\, e_k(0) \qquad 1 \leq k \leq n \tag{7-8-5}$$

Clearly, $e(t) \rightarrow 0$ as $t \rightarrow \infty$. Not only does the error go to zero by "sliding" down the switching surface, but the *rate* at which the error decreases can be controlled through the specification of the gain F, which controls the "slope" of the surface. Since Eq. (7-8-5) is totally independent of the robotic arm parameters, the variable-structure control system is robust when it is operating in the sliding mode.

There remains the problem of developing a control law $\tau = f(x)$ which will ensure that the system operates in the sliding mode. To develop a suitable control law, we make use of Liapunov techniques. Consider, in particular, the following function:

$$V_L(x) = \frac{\sigma^T(x)\sigma(x)}{2} \tag{7-8-6}$$

This is a Liapunov-type function in the sense that $V_L(x)$ is continuously differentiable, $V_L(x) \geq 0$, and $V_L(x) = 0$ if and only if $\sigma(x) = 0$. To show that the solution of the closed-loop system approaches the switching surface, it is sufficient to show that, along solutions of Eq. (7-8-1), $\dot{V}_L(x(t)) \leq 0$ and $\dot{V}_L(x(t)) \equiv 0$ implies $\sigma(x(t)) \equiv 0$, where $\dot{V}_L(x) = \sigma^T(x)\dot{\sigma}(x)$. However, this would only guarantee that the solution approaches the switching surface in the limit as $t \rightarrow \infty$. To ensure that the solution of the closed-loop system hits the switching surface in a finite time, we use the following, somewhat stronger condition.

Proposition 7-8-1: Sliding Mode. Let $\sigma(x)$ be the switching surface in Eq. (7-8-2). If there exists a $\gamma > 0$ such that $\sigma^T(x)\dot{\sigma}(x) \leq -\gamma \| \sigma(x) \|$ along solutions of Eq. (7-8-1), then $x(t)$ will hit the switching surface at time t_{switch}, where:

$$t_{\text{switch}} \leq \frac{\| \sigma(x(0)) \|}{\gamma}$$

Proof. Suppose there is a $\gamma > 0$ such that $\sigma^T(x)\sigma(x) \leq -\gamma \| \sigma(x) \|$ along solutions of Eq. (7-8-1). Define the scalar $z \triangleq \| \sigma(x) \|$. Then:

$$\begin{aligned} z\dot{z} &= \frac{d}{dt}\frac{z^2}{2} \\ &= \frac{d}{dt}\frac{\| \sigma(x) \|^2}{2} \\ &= \frac{d}{dt}\frac{\sigma^T(x)\sigma(x)}{2} \end{aligned}$$

$$= \sigma^T(x)\dot{\sigma}(x)$$
$$\leq -\gamma \| \sigma(x) \|$$
$$= -\gamma z$$

When x is off the switching surface, $z > 0$. Dividing both sides of the last equation by z and integrating from 0 to t yields:

$$z(t) \leq z(0) - \gamma t$$

The solution $x(t)$ hits the switching surface when $z(t) = 0$. Setting $z(t) = 0$ and solving for t_{switch} yields:

$$t_{\text{switch}} \leq \frac{z(0)}{\gamma}$$
$$= \frac{\| \sigma(x(0)) \|}{\gamma}$$

Thus if $V_L(x(t))$ decreases at a rate proportional to $\| \sigma(x(t)) \|$, the solution of Eq. (7-8-1) will strike the switching surface within a time that is bounded from above by t_{switch}. Of course, $x(0)$ may already be on the switching surface, since the reference trajectory $r(t)$ is often chosen to start from the current robot position. In any event, once $x(t)$ reaches the switching surface, the dynamics that brought it there will also keep it there. Indeed, as soon as $x(t)$ penetrates the switching surface, the control law switches and thereby directs the solution back toward the switching surface. This process continues as the solution "zigzags" back and forth across the switching surface, as shown in Fig. 7-33. Since $\dot{e} = -Fe$ on the switching surface, the solution tends to slide down the surface toward the origin as it switches back and forth. This is the so-called sliding mode.

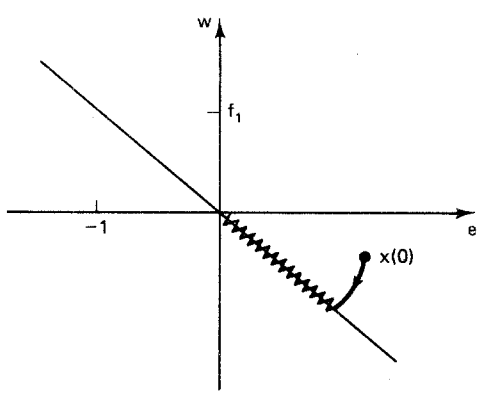


Figure 7-33 Sliding-mode solution.

In theory, the oscillations about the switching surface have zero amplitude and infinite frequency. However, in practice they have a small amplitude and a high frequency depending on the relative values of F and γ and the maximum switching speed of the control law.

The key to applying Prop. 7-8-1 is finding a control law $\tau = f(x)$ such that $\sigma^T(x)\sigma(x) \leq -\gamma\|\sigma(x)\|$ for some $\gamma > 0$. For an n-axis robot, this is a challenging task (Young, 1978). To illustrate the derivation of a control law, we examine a simple special case, the one-axis robot, or inverted pendulum, in Fig. 7-2. From Eq. (7-2-2), the closed-loop state-space model of the one-axis robot, in terms of $x = [e, \dot{e}]^T$, is:

$$\dot{x}_1 = x_2 \tag{7-8-7a}$$

$$\dot{x}_2 = \ddot{r} - \frac{\tau - \alpha(r - x_1) - \beta(\dot{r} - x_2)}{\delta} \tag{7-8-7b}$$

$$\tau = f(x) \tag{7-8-7c}$$

Here $\alpha(q)$, $\beta(v)$, and δ denote the gravitational force, frictional force, and moment of inertia, respectively. That is,

$$\alpha(q) = g_0\left(\frac{m_1}{2} + m_L\right)a_1 \cos q \tag{7-8-8a}$$

$$\beta(v) = b_1^v v + \operatorname{sgn}(v)(b_1^d + (b_1^s - b_1^d)\exp(-|v|/\epsilon)) \tag{7-8-8b}$$

$$\delta = \left(\frac{m_1}{3} + m_L\right)a_1^2 \tag{7-8-8c}$$

To develop a control law we must first establish bounds on the values of the robot parameters, including the rate of change of the reference trajectory. Suppose:

$$|\alpha(q)| < \alpha_1 \tag{7-8-9a}$$

$$|\beta(v)| < \beta_0 + \beta_1|v| \tag{7-8-9b}$$

$$0 < \delta_0 < \delta < \delta_1 \tag{7-8-9c}$$

$$|\dot{r}| < r_1 \quad , \quad |\ddot{r}| < r_2 \tag{7-8-9d}$$

Thus upper bounds are placed on the magnitudes of $\alpha(q)$, $\beta(v)$, and $\dot{r}$, and positive lower and upper bounds are placed on the moment of inertia δ. Given Eq. (7-8-2) and inequalities (7-8-9), we can now evaluate $\dot{V}_L(x) = \sigma(x)\dot{\sigma}(x)$ along solutions of (7-8-7):

$$\begin{aligned}
\sigma(x)\dot{\sigma}(x) &= \sigma(x)[f_1\dot{x}_1 + \dot{x}_2] \\
&= \sigma(x)\left[f_1 x_2 + \ddot{r} - \frac{\tau - \alpha(r - x_1) - \beta(\dot{r} - x_2)}{\delta}\right] \\
&= -\frac{\sigma(x)\tau}{\delta} + \sigma(x)\left[f_1 x_2 + \ddot{r} + \frac{\alpha(r - x_1)}{\delta} + \frac{\beta(\dot{r} - x_2)}{\delta}\right] \\
&\leq -\frac{\sigma(x)\tau}{\delta} + \left|\sigma(x)\left[f_1 x_2 + \ddot{r} + \frac{\alpha(r - x_1)}{\delta} + \frac{\beta(\dot{r} - x_2)}{\delta}\right]\right| \\
&= -\frac{\sigma(x)\tau}{\delta} + |\sigma(x)| \cdot \left|\left[f_1 x_2 + \ddot{r} + \frac{\alpha(r - x_1)}{\delta} + \frac{\beta(\dot{r} - x_2)}{\delta}\right]\right|
\end{aligned}$$

$$\leq -\frac{\sigma(x)\tau}{\delta} + |\sigma(x)|\left[|[f_1 x_2| + |\ddot{r}| + \left|\frac{\alpha(r - x_1)}{\delta}\right| + \left|\frac{\beta(\dot{r} - x_2)}{\delta}\right|\right]$$

$$= -\frac{\sigma(x)\tau}{\delta} + |\sigma(x)|\left[f_1|x_2| + |\ddot{r}| + \frac{|\alpha(r - x_1)|}{\delta} + \frac{|\beta(\dot{r} - x_2)|}{\delta}\right]$$

$$< -\frac{\sigma(x)\tau}{\delta} + |\sigma(x)|\left[f_1|x_2| + r_2 + \frac{\alpha_1}{\delta_0} + \frac{(\beta_0 + \beta_1|\dot{r} - x_2|}{\delta_0}\right]$$

$$= -\frac{\sigma(x)\tau}{\delta} + |\sigma(x)|\left[f_1|x_2| + r_2 + \frac{(\alpha_1 + \beta_0)}{\delta_0} + \frac{\beta_1|\dot{r} - x_2|}{\delta_0}\right]$$

$$\leq -\frac{\sigma(x)\tau}{\delta} + |\sigma(x)|\left[f_1|x_2| + r_2 + \frac{(\alpha_1 + \beta_0)}{\delta_0} + \frac{\beta_1(|\dot{r}| + |x_2|)}{\delta_0}\right]$$

$$< -\frac{\sigma(x)\tau}{\delta} + |\sigma(x)|\left[f_1|x_2| + r_2 + \frac{(\alpha_1 + \beta_0)}{\delta_0} + \frac{\beta_1(r_1 + |x_2|)}{\delta_0}\right]$$

$$= -\frac{\sigma(x)\tau}{\delta} + |\sigma(x)|\left[\frac{(\alpha_1 + \beta_0 + \beta_1 r_1 + \delta_0 r_2)}{\delta_0} + \frac{(\beta_1 + \delta_0 f_1)|x_2|}{\delta_0}\right]$$

$$= -\frac{\sigma(x)\tau}{\delta} + \frac{(\alpha_1 + \beta_0 + \beta_1 r_1 + \delta_0 r_2)}{\delta_0}|\sigma(x)| + \frac{\beta_1 + \delta_0 f_1}{\delta_0}|\sigma(x)x_2|$$

The control law $\tau = f(x)$ must be selected in such a way that it "dominates" the two positive terms in Eq. (7-8-10). As a candidate, consider the following variable-structure control law:

$$\tau = k_1 \operatorname{sgn}[\sigma(x)] + k_2 \operatorname{sgn}[\sigma(x)x_2]\, x_2 \tag{7-8-11}$$

Here $k_1 > 0$ and $k_2 > 0$ are controller gains that remain to be determined. Substituting Eq. (7-8-11) in Eq. (7-8-10) and using the identity $\operatorname{sgn}(z)\, z = |z|$ yields:

$$\sigma(x)\dot{\sigma}(x) < -\frac{\sigma(x)\tau}{\delta} + \frac{(\alpha_1 + \beta_0 + \beta_1 r_1 + \delta_0 r_2)|\sigma(x)| + (\delta_0 f_1 + \beta_1)|\sigma(x)x_2|}{\delta_0}$$

$$= -\sigma(x)\frac{k_1 \operatorname{sgn}[\sigma(x)] + k_2 \operatorname{sgn}[\sigma(x)x_2]\, x_2}{\delta} + \frac{(\alpha_1 + \beta_0 + \beta_1 r_1 + \delta_0 r_2)|\sigma(x)| + (\delta_0 f_1 + \beta_1)|\sigma(x)x_2|}{\delta_0}$$

$$= -\frac{k_1|\sigma(x)| + k_2|\sigma(x)x_2|}{\delta} + \frac{(\alpha_1 + \beta_0 + \beta_1 r_1 + \delta_0 r_2)|\sigma(x)| + (\delta_0 f_1 + \beta_1)|\sigma(x)x_2|}{\delta_0}$$

$$< -\frac{k_1|\sigma(x)| + k_2|\sigma(x)x_2|}{\delta_1}$$

$$+\frac{(\alpha_1+\beta_0+\beta_1 r_1+\delta_0 r_2)|\sigma(x)|+(\delta_0 f_1+\beta_1)|\sigma(x)x_2|}{\delta_0}$$

$$=\left(\frac{\alpha_1+\beta_0+\beta_1 r_1+\delta_0 r_2}{\delta_0}-\frac{k_1}{\delta_1}\right)|\sigma(x)|$$

$$+\left(\frac{\delta_0 f_1+\beta_1}{\delta_0}-\frac{k_2}{\delta_1}\right)|\sigma(x)x_2| \tag{7-8-12}$$

It is clear from expression (7-8-12) that $\sigma(x)\dot{\sigma}(x)$ can be made negative if the controller gains k_1 and k_2 are sufficiently large, that is, if

$$k_1 > \frac{\delta_1(\alpha_1+\beta_0+\beta_1 r_1+\delta_0 r_2)}{\delta_0} \tag{7-8-13a}$$

$$k_2 > \frac{\delta_1(\delta_0 f_1+\beta_1)}{\delta_0} \tag{7-8-13b}$$

Note that the required gains increase as the uncertainty in the moment of inertia, δ_1/δ_0, increases. The constraint on k_2 ensures that the second term in expression (7-8-12) will be negative, which means it can be neglected without violating the inequality. This results in the following simplified inequality, which satisfies the hypothesis of Prop. 7-8-1:

$$\sigma(x)\dot{\sigma}(x) < -\gamma|\sigma(x)| \tag{7-8-14}$$

where

$$\gamma = \frac{k_1}{\delta_1}-\frac{(\alpha_1+\beta_0+\beta_1 r_1+\delta_0 r_2)}{\delta_0} \tag{7-8-15}$$

Thus if the controller gains satisfy inequalities (7-8-13), then the variable-structure control law in Eq. (7-8-11) will drive the one-axis robot to the switching surface $f_1 x_1 + x_2 = 0$ and then keep it there. Consequently, $x_1(t)$ will track $r(t)$ with a performance that is independent of the robotic arm parameters as long as the parameters satisfy inequalities (7-8-9).

Example 7-8-1: One-Axis Robot

Consider the controller gains for the one-axis robot in Fig. 7-2. Suppose the link mass m_1, load mass m_L, link length a_1, coefficients of friction $\{b_1^v, b_1^d, b_1^s\}$ lie within the following numerical ranges:

$$2.8 \le m_1 \le 3.2$$

$$0.0 \le m_L \le 1.0$$

$$0.9 \le a_1 \le 1.1$$

$$0.0 \le b_1^v \le 1.0$$

$$0.0 \le b_1^d \le 1.0$$

$$0.0 \le b_1^s \le 5.0$$

Notice that the bounds on m_L dictate that the robot can be operating anywhere

from a no load condition ($m_L = 0$) to a full load condition ($m_L = 1$), which corresponds to a load at the end of the arm of about one-third of the robot mass. In addition, the bounds on b_1^v, b_1^d and b_1^s include the possibility of frictionless operation. Using $g_0 = 9.81$ and Eq. (7-8-8), we can derive values for the bounds in inequalities (7-8-9) as follows:

$$\alpha_1 = 9.81\left(\frac{3.2}{2} + 1.0\right)(1.1) = 28.1$$

$$\beta_0 = 5.0$$

$$\beta_1 = 1.0$$

$$\delta_0 = \left(\frac{2.8}{3} + 0.0\right)(0.9)^2 = 0.76$$

$$\delta_1 = \left(\frac{3.2}{3} + 1.0\right)(1.1)^2 = 2.50$$

Suppose the reference trajectory is $r(t) = \pi[1 - \exp(-t/2)]$ for $t \geq 0$. Then $\dot{r}(t) = \pi[\exp(-t/2)]/2$, and $\ddot{r}(t) = -\pi[\exp(-t/2)]/4$. Hence:

$$r_1 = \frac{\pi}{2} = 1.57 \quad , \quad r_2 = \frac{\pi}{4} = 0.79$$

To determine the constraints on the controller gains, we have to first choose the slope of the switching surface. Suppose the sliding-mode gain is $f_1 = 2.0$. Then, from inequalities (7-8-13), the constraints on the variable-structure controller gains are:

$$k_1 > \frac{2.50[28.1 + 5.0 + 1.0(1.57) + 0.76(0.79)]}{0.76} = 116.0$$

$$k_2 > \frac{2.50[0.76(2.0) + 1.0]}{0.76} = 8.3$$

To account for round-off error, and to ensure that γ in Eq. (7-8-15) is sufficiently large, we choose values somewhat larger than the minimum. For example, one possible variable-structure control law for the one-axis robot is:

$$\tau = -\{125 \operatorname{sgn}(2x_1 + x_2) + 10 \operatorname{sgn}[(2x_1 + x_2)x_2]\, x_2\}$$

Next, we estimate the time it takes to reach the switching surface. Suppose the link starts out motionless in the straight-down position, $x(0) = [-\pi/2, 0]^T$. From Eq. (7-8-15), the rate constant γ is:

$$\gamma = \frac{125}{2.50} - \frac{28.1 + 5.0 + 1.0(1.57) + 0.76(0.79)}{0.76} = 3.59$$

Thus, from Prop. 7-8-1, the one-axis robot in this case will hit the switching surface at time t_{switch} where :

$$t_{\text{switch}} < \frac{|2.0(1.57) + 0|}{3.59} = 0.87$$

Once the robot reaches the switching surface, it will enter the sliding mode, where $\dot{e} = -2e$. In the sliding mode, the tracking error approaches zero exponentially with a time constant of $1/f_1 = 0.5$ sec.

The variable-structure control method is a robust method that appears to be well suited for robotic manipulators because it requires only *bounds* on the robotic arm parameters. However, the variable-structure control method does have its drawbacks. One is that there is no single systematic procedure that is guaranteed to produce a suitable control law. Another drawback is the fact that the control signal τ is often quite large and must switch values rapidly when the system is operating in the sliding mode. To reduce this "chattering" of the control signal, the sgn function in the control law can be replaced by a continuous approximation with a large slope. This has the effect of (gradually) switching the controller gains in a small band around the switching surface rather than right at the surface (Asada and Slotine, 1986).

7-9 IMPEDANCE CONTROL

The control techniques discussed thus far are examples of position control methods—methods that assume that the robot is moving freely in the workspace. It is also possible to design controllers that operate when the tool is in contact with the environment (Salisbury, 1980; Hogan, 1985; Asada and Slotine, 1986; Whitney, 1987; An et al., 1988). This is an area of active research and is referred to by terms such as compliance control, force control, hybrid control, and impedance control. In this section, we examine one specific approach for illustration purposes, namely, impedance control.

Each manipulation task has associated with it a set of *natural* constraints which are imposed on the robot by the environment (Mason, 1981). For example, suppose the tool tip comes into contact with a hard surface G as shown in Fig. 7-34. Here the environment imposes a natural position constraint which prevents the tool tip from penetrating the surface G. When the tool is in contact with G, it no longer makes sense to attempt to control *position* in the direction d orthogonal to G. However, we can control tool-tip *force* along d, for example, to maintain contact with the surface. Furthermore, we can simultaneously control the tool-tip position along directions tangent to G. In this way the tool can be made to slide along the surface.

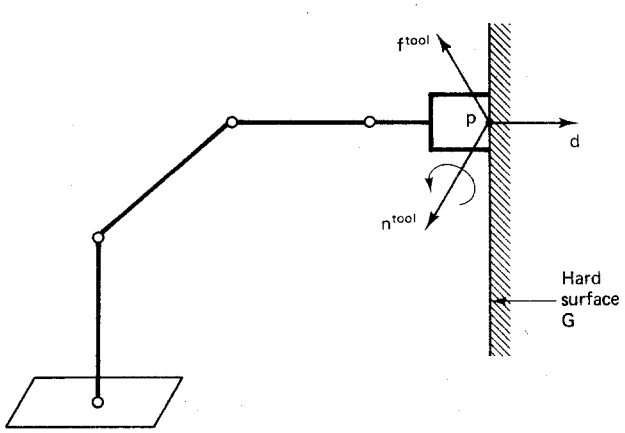


Figure 7-34 Natural position constraint imposed by environment.

This is called a *compliant* move along the surface G, since the trajectory is made to "comply" with the natural constraints imposed by G.

Recall from the static analysis in Prop. 5-8-1 that an external end-of-arm force and moment vector F^{tool} induces a torque τ at the robot joints as follows:

$$\tau = J^T(q)F^{\text{tool}} \tag{7-9-1}$$

Here $J(q)$ is the $6 \times n$ manipulator Jacobian matrix which relates an infinitesimal joint displacement dq to the corresponding infinitesimal tool displacement du:

$$du = J(q)\, dq \tag{7-9-2}$$

Suppose we want to design a controller that generates a specified end-of-arm "stiffness," which can be modeled by the following generalized spring equation:

$$F^{\text{tool}} = K\, du \tag{7-9-3}$$

Here du represents an infinitesimal displacement between the desired tool position and orientation and the measured tool position and orientation, and K is a positive-definite *spring constant* matrix. For example, K might be diagonal with positive diagonal elements:

$$K = \text{diag}\,\{k_1, k_2, \ldots, k_6\} \tag{7-9-4}$$

The scalar k_i denotes the stiffness along dimension i. For directions along which position must be controlled, a *large* value can be assigned to k_i. This has the effect of severely penalizing small displacements in position. For directions in which the tool force or moment is to be controlled, smaller values can be assigned to k_i. The tool force or moment is controlled only *indirectly*, because from Eq. (7-9-3) we see that it is actually the *relationship* between force and displacement, the stiffness or mechanical "impedance" of the arm, that is being controlled. Consequently, control techniques which use Eq. (7-9-3) as their objective are referred to as *impedance control* methods.

Let e denote the difference between the desired tool position and orientation r and the actual position and orientation u. That is, e represents the *error*, or deflection of the tool from its reference position and orientation:

$$e \triangleq r - u \tag{7-9-5}$$

If we assume that the error e plays the role of du in Eq. (7-9-2), then, combining Eqs. (7-9-1) and (7-9-2), we see that the joint torque required to maintain a desired end-of-arm stiffness is $\tau = J^T(q)Ke$. As a generalization of this formulation, consider the following control law, which also includes damping and the effects of gravity (Asada and Slotine, 1986):

$$\tau = J^T(q)[Ke + L\dot{e}] + h(q) \tag{7-9-6}$$

Here L is a positive-definite diagonal *damping constant* matrix, and $h(q)$ represents the load torque due to gravity. Note the similarity between the impedance control law in Eq. (7-9-6) and the PD-plus-gravity control law in Eq. (7-6-2). The basic difference here is that the error e in Eq. (7-9-6) is specified directly in terms of the tool position and orientation; it does not represent joint position error. In addition, the

diagonal components of K and L can vary substantially in order to achieve stiffness and damping control in some directions and position control in other directions. Refer to Fig. 7-35 for a block diagram representation of an impedance control system for a robotic arm.

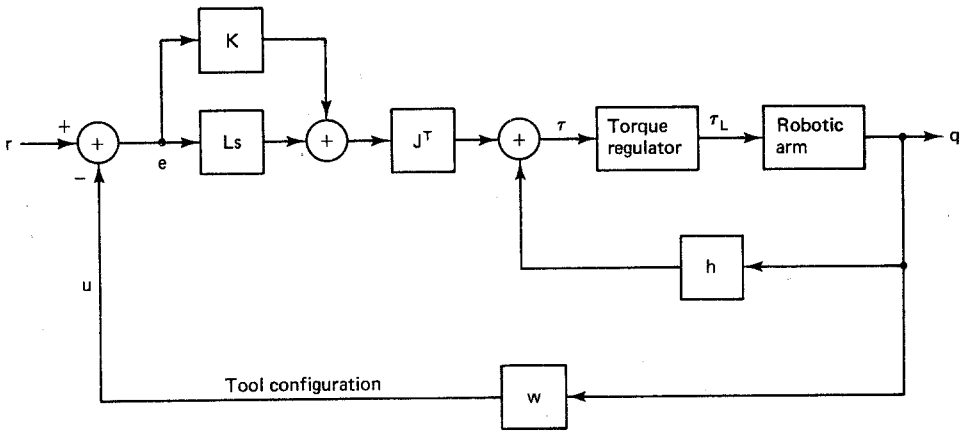


Figure 7-35 Impedance control of a robotic arm.

Since an impedance controller is generally similar to a PD-plus-gravity controller, we can analyze impedance control using Liapunov techniques. However, the analysis is complicated by the fact that the position and velocity gain matrices include $J(q)$, which can lose rank if q is at a joint-space singularity. To examine the equilibrium points of the impedance control system in Fig. 7-35, suppose the reference position and orientation of the tool is a constant $r(t) = u^R$, where u^R lies "inside" an environmental object as shown in Fig. 7-36. Here the environment is deformed in the vicinity of the point of contact, and this deformation creates an external end-of-arm reaction force and moment F^{tool}. If the environmental surface is modeled as a generalized spring, then:

$$F^{\text{tool}} = H(u^E - u) \tag{7-9-7}$$

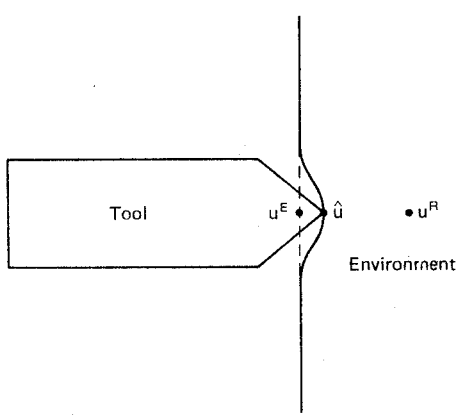


Figure 7-36 Deformation of environment by robot.

In this case u^E represents the initial point of contact before the deformation takes place, and H is a positive-definite matrix which specifies the spring constant or stiffness of the environment. Note that if u^R does not lie inside the environmental object, then $u^E = u$ in Eq. (7-9-7), which means that the reaction force and moment is zero. The reaction force and moment from the environment can be included as an additional component of the generalized force acting on the arm. In this case, the dynamic model of the arm developed in Eq. (6-4-9) includes a new term, as follows:

$$D(q)\ddot{q} + c(q, \dot{q}) + h(q) + b(\dot{q}) = \tau + J^T(q)F^{\text{tool}} \tag{7-9-8}$$

Note that the transpose of the Jacobian matrix is included to convert the end-of-arm force and moment vector to an equivalent joint torque using Prop. 5-8-1. To develop the closed-loop equations of the impedance controller, we convert Eq. (7-9-8) to state-space form using Eqs. (7-9-6) and (7-9-7). Define the state vector as $x^T = [q^T, v^T]$, where $v = \dot{q}$. Then,

$$\dot{q} = v \tag{7-9-9a}$$

$$\dot{v} = D^{-1}(q)\{J^T(q)[K(r - u) + L(\dot{r} - \dot{u}) + H(u^E - u)] - c(q, v) - b(v)\} \tag{7-9-9b}$$

We find the equilibrium points of the closed-loop system by setting the right-hand side of Eq. (7-9-9) to zero, assuming $r(t) = u^R$ for $t \geq 0$, where u^R is the tool set point. This yields $\dot{r} = 0$ and $v = 0$, which in turn means $c(q, 0) = 0$ and $b(0) = 0$. In addition, at an equilibrium point the tool will not be moving, which implies that $\dot{u} = 0$. Therefore the equilibrium points of the closed-loop impedance control system are solutions of the following equation:

$$J^T(q)[K(u^R - u) + H(u^E - u)] = 0 \tag{7-9-10}$$

Recall that $J(q)$ is the $6 \times n$ manipulator Jacobian matrix. If $n \geq 6$ and q is not a joint-space singularity, then rank $[J(q)] = 6$. In this case, the 6×6 matrix $J(q)J^T(q)$ is nonsingular. Thus we can multiply both sides of Eq. (7-9-10) by $[J(q)J^T(q)]^{-1}J(q)$, and the result is $K(u^R - u) + H(u^E - u) = 0$. Collecting the coefficients of u then yields $Ku^R + Hu^E - (K + H)u = 0$. Therefore, if $n \geq 6$ and q is not a joint-space singularity, the closed-loop impedance control system has a single equilibrium point at:

$$\hat{u} = (K + H)^{-1}(Ku^R + Hu^E) \tag{7-9-11}$$

Note that if the controller spring is soft in comparison with the environment ($k_i \ll h_i$, $1 \leq i \leq 6$), then $\hat{u} \approx u^E$, which means there is very little deformation of the environment. Alternatively, if the controller spring is stiff in comparison with the environment ($k_i \gg h_i$, $1 \leq i \leq 6$), then $\hat{u} \approx u^R$, which means there is very little deflection of the tool. In this case the robot is operating in a position control mode. More generally, along some dimensions position control can be used, whereas along others impedance control can be employed. The diagonal components of the matrix L are typically selected to provide critical damping.

By using Liapunov's second method it is possible to show that the equilibrium point $\hat{u}$ in Eq. (7-9-11) is asymptotically stable. The proof is analogous to the proof of Prop. 7-6-1, but is more complex (Asada and Slotine, 1986). We conclude our discussion of impedance control with an example.

Example 7-9-1: Impedance Control

Consider the three-axis SCARA robot in Fig. 7-4. To derive an impedance control law, we need to know the gravity loading $h(q)$ and the manipulator Jacobian $J(q)$. From Eq. (7-2-10), the gravity loading vector is simply $h(q) = [0, 0, -m_3 g_0]^T$. The manipulator Jacobian of a four-axis SCARA robot was derived in Eq. (5-7-19). If we follow the same procedure but ignore the last joint, the tool roll joint, this results in the following manipulator Jacobian for the three-axis case:

$$J(q) = \left[\begin{array}{ccc} -a_1 S_1 - a_2 S_{1-2} & a_2 S_{1-2} & 0 \\ a_1 C_1 + a_2 C_{1-2} & -a_2 C_{1-2} & 0 \\ 0 & 0 & -1 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & -1 & 0 \end{array}\right]$$

The three-axis SCARA robot can place the tool tip at arbitrary positions in three-dimensional space, but it cannot control the tool orientation. Consequently, we will assume that $u = p$, which means that we will attempt to control only the position part of the tool position and orientation. Therefore, only the 3×3 upper block of $J(q)$ is used. Suppose the controller spring constant matrix K, controller damping constant matrix L, and environment spring constant matrix H are all diagonal matrices, as follows:

$$K = \text{diag}\{k_1, k_2, k_3\}$$

$$L = \text{diag}\{l_1, l_2, l_3\}$$

$$H = \text{diag}\{h_1, h_2, h_3\}$$

Applying Eq. (7-9-6), the control law of an impedance controller for the three-axis SCARA robot is:

$$e = r - p$$

$$\tau_1 = -(a_1 S_1 + a_2 S_{1-2})(k_1 e_1 + l_1 \dot{e}_1) + (a_1 C_1 + a_2 C_{1-2})(k_2 e_2 + l_2 \dot{e}_2)$$

$$\tau_2 = a_2 S_{1-2}(k_1 e_1 + l_1 \dot{e}_1) - a_2 C_{1-2}(k_2 e_2 + l_2 \dot{e}_2)$$

$$\tau_3 = -(k_3 e_3 + l_3 \dot{e}_3) - m_3 g_0$$

The controller gains K and L must be selected on the basis of the manipulation task. Suppose the robot tool is a pen or marker and the task is to scribe a circle on a "lumpy" mat secured to the work surface, as shown in Fig. 7-37. Assume the circle has radius ρ and is in the $x^0 y^0$ plane centered at point (a, b). If the circle is to be traced out in time T, a nominal reference trajectory is:

$$r(t) = \left[a + \rho \cos\frac{2\pi t}{T},\ b + \rho \sin\frac{2\pi t}{T},\ 0\right]^T \qquad 0 \le t \le T$$

This is a three-dimensional manipulation task characterized by two artificial constraints and one natural constraint. The two artificial constraints are $p_1(t) = r_1(t)$ and $p_2(t) = r_2(t)$. The natural constraint is $p_3(t) = \sigma(t)$, where $\sigma(t)$ is the *unknown* height of the mat, which varies as the circle is traversed. For the manipulation task to be successful, the tool tip must follow the terrain of the mat so as to maintain contact as the x

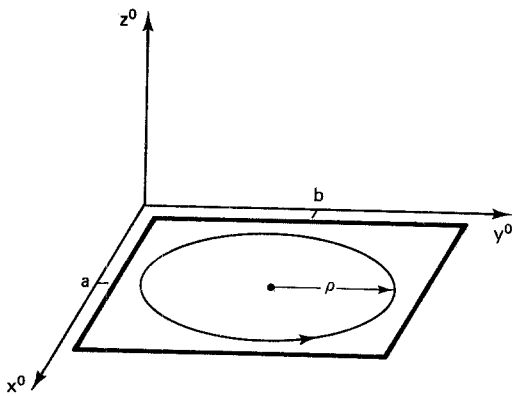


Figure 7-37 Scribing a circle on a lumpy mat.

and y tool coordinates trace out the circle. Thus we need position control along the x and y dimensions and impedance control along the z dimension. This dictates the following constraints on the controller spring constants:

$$k_1 \gg h_1$$

$$k_2 \gg h_2$$

$$k_3 \approx h_3$$

If $k_3 = h_3$, where h_3 models the stiffness of the mat, then, from Eq. (7-9-11), we see that the tool will compress the mat by 50 percent as it traces the circle. Clearly, the compression factor can be modified by changing the value of k_3. The controller damping constants must also be selected. To achieve critical damping ($\zeta_i = 1$), we use $l_i = 2(k_i)^{1/2}$ for $1 \leq i \leq 3$.

7-10 PROBLEMS

7-1. Phase-locked loop (PLL) circuits are used in a variety of control and communications applications. A PLL can be modeled using the following second-order nonlinear equation, where y represents the phase angle between the input signal and an internal voltage-controlled oscillator:

$$\ddot{y} + (a + b \cos y)\dot{y} + c \sin y = 0$$

Find the state equations of this system if $x = [y, \dot{y}]^T$.

7-2. Consider the PLL system in Prob. 7-1 with $a > b > 0$ and $c > 0$. Find the equilibrium points.

7-3. Consider the PLL system in Prob. 7-1 with $a > b > 0$ and $c > 0$. Analyze the stability of the equilibrium point at $x = 0$, using Liapunov's first method.

7-4. Consider the PLL system in Prob. 7-1 with $a > b > 0$ and $c > 0$. Analyze the stability of the equilibrium point at $x = 0$, using Liapunov's second method, and find the domain of attraction Ω. *Hint:* Try $V_L(x) = \alpha(1 - \cos x_1) + \beta x_2^2$.

7-5. Consider the following two-dimensional nonlinear system, where $u(t) = a$ for some constant a:

$$\dot{x}_1 = -x_2 + x_1(x_1^2 + x_2^2 - u_1)$$

$$\dot{x}_2 = x_1 + x_2(x_1^2 + x_2^2 - u_1)$$

Use Liapunov's second method to show that $x = 0$ is asymptotically stable when $a > 0$. Estimate the domain of attraction Ω.

7-6. Find a PD-gravity control law for the one-axis robot in Fig. 7-2.

7-7. Find a computed-torque control law for the one-axis robot in Fig. 7-2, assuming the parameter estimates are exact.

7-8. Find an impedance control law for the one-axis robot in Fig. 7-2, assuming that K and L are diagonal and $u = p \in \mathbf{R}^2$.

7-9. Find an impedance control law of the two-axis planar articulated robot in Fig. 7-3, assuming that K and L are diagonal and $u = p \in \mathbf{R}^2$.

7-10. Consider the three-axis spherical-coordinate robot shown in Fig. 7-38. Suppose that the tool mass is small in comparison with m_2 and that $q_2 = 0$ in the position shown. Find the gravity loading vector $h(q)$, assuming $g = [0, 0, -g_0]^T$.

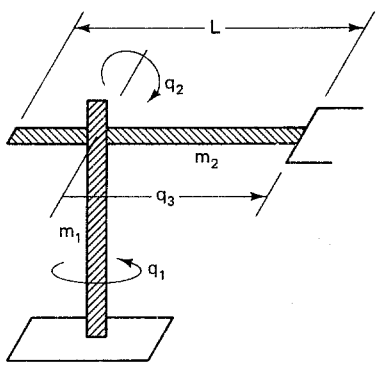


Figure 7-38 A three-axis spherical-coordinate robot.

7.11. Find a PD-gravity control law for the three-axis spherical-coordinate robot in Fig. 7-38, assuming that K and L are diagonal.

7-12. Consider the three-axis cylindrical-coordinate robotic arm shown in Fig. 7-39. Here m_2 and m_T translate up and down along axis 2. Find the gravity loading vector $h(q)$, assuming $g = [0, 0, -g_0]^T$.

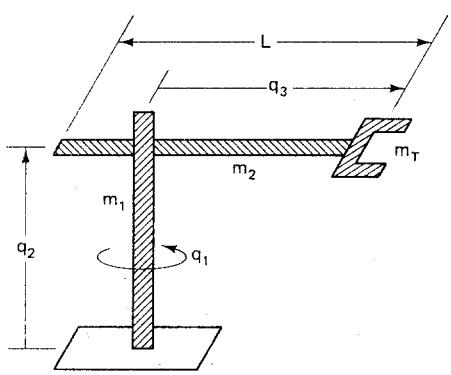


Figure 7-39 A three-axis cylindrical-coordinate robot.

7.13. Find a PD-gravity control law for the three-axis cylindrical-coordinate robot in Fig. 7-39, assuming that K and L are diagonal.

7-14. Consider the speed-regulator feedback system shown in Fig. 7-40. Here the input $u(t)$

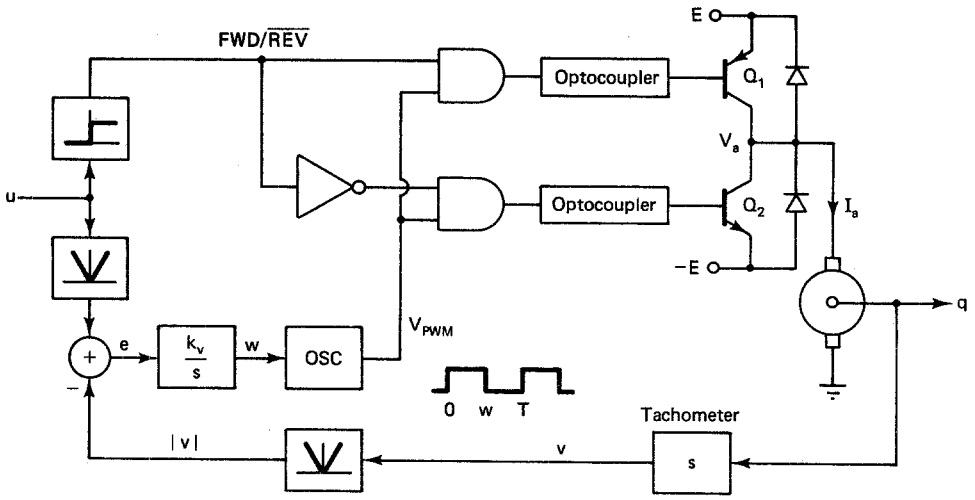


Figure 7-40 Speed-regulator system.

represents the desired speed and $v(t) = \dot{q}(t)$ represents the measured speed of the load shaft. Using a method similar to that used for the torque regulator in Sec. 7-5-2, find the simplified transfer function ($L_a = 0$) of the speed regulator. That is, find:

$$G_s(s) \triangleq \frac{V(s)}{U(s)}$$

7-15. For the speed-regulator system in Fig. 7-40, for what values of the gain k_v is the closed-loop system stable?

7-16. Find the steady-state tracking error of the speed-regulator system in Fig. 7-40 for the following inputs, where $1(t)$ denotes the unit step:
(a) $u(t) = 1(t)$
(b) $u(t) = t\,1(t)$

7-17. Consider the following state-space model of a linear time-invariant system with $x(0) = 0$:

$$\dot{x}(t) = Ax(t) + bu(t)$$
$$y(t) = c^T x(t) + du(t)$$

Here $x(t) \in \mathbf{R}^n$ is the state of the system at time t. Use the fact that the Laplace transform of $\dot{x}(t)$ is $sX(s) - x(0)$ to show that the transfer function $G(s) = Y(s)/U(s)$ is:

$$G(s) = c^T[sI - A]^{-1}b + d$$

7-18. Consider a linear system with the following transfer function:

$$G(s) = \frac{k_m}{s(T_m s + 1)}$$

Is this system stable? If not, find a bounded input $u(t)$ which generates an unbounded output $y(t)$. *Hint:* Pick a $u(t)$ such that $Y(s)$ has a double pole on the imaginary axis.

7-19. Consider the following two-dimensional nonlinear system with input $u(t)$:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = u - g(x_1, x_2)$$

Suppose there exist constants $\alpha > 0$, $\beta > 0$, and $\delta > 0$ such that:

$$|g(x_1, x_2)| < \alpha + \beta|x_1| + \delta|x_2|$$

Using a switching surface gain of 3, find a variable-structure controller $u = f(x)$ for this system. If $x(0) = [1, 0]^T$, how long will it take for $x(t)$ to hit the switching surface?

REFERENCES

AN, C. H., C. G. ATKESIN, AND J. M. HOLLERBACH (1988). *Model-Based Control of a Robot Manipulator*, MIT Press: Cambridge, Mass.

ASADA, H., AND J. E. SLOTINE (1986). *Robot Analysis and Control*, Wiley: New York.

BEJCZY, A. K. (1974). "Robot arm dynamics and control," *Tech. Memo. 33-669*, Jet Propulsion Laboratory, Pasadena, Calif.

DUBOWSKI, S., AND D. T. DESFORGES (1979). "The application of model-referenced adaptive control to robotic manipulators," *J. Dynamic Systems, Measurement, and Control*, Vol. 101, pp. 193–200.

HENDRICKSON, T., AND H. SANDHU (1985). *XR-3 Robot Arm and Mark III 8 Axis Controller Owner's Manual*, Rhino Robots, Inc: Champaign, Ill.

HOGAN, N. (1985) "Impedance control: An approach to manipulation," *J. Dynamic, Systems, Measurement and Control*, Vol. 107, pp. 1–24.

ITKIS, U. (1976). *Control Systems of Variable Structure*, Wiley: New York.

KODITSCHEK, D. E. (1984). "Natural motion for robot arms," *Proc. 23rd IEEE Conf. Decision and Control*, Las Vegas, December, pp. 733–735.

KOIVO, A. J., AND T. H. GUO (1983). "Adaptive linear controller for robotic manipulators," *IEEE Trans. Automatic Control*, Vol. AC-28, pp. 162–171.

KUO, B. C. (1982). *Automatic Control Systems*, Prentice Hall: Englewood Cliffs, N.J.

LEE, C. S. G., AND M. J. CHUNG (1984). "An adaptive control strategy for mechanical manipulators," *IEEE Trans. Automatic Control*, Vol. AC-29, pp. 837–840.

LUH, J. Y. S., M. W. WALKER, AND R. P. C. PAUL (1980). "On-line computational scheme for mechanical manipulators," *J. Dynamic Systems, Measurement and Control*, Vol. 102, pp. 69–76.

MASON, M. T. (1981). "Compliance and force control for computer controller manipulators," *IEEE Trans. Syst., Man, Cybern.*, Vol. SMC-11, pp. 418–432.

OGATA, K. (1970). *Modern Control Engineering*, Prentice Hall: Englewood, Cliffs, N.J.

SALISBURY, J. K. (1980), "Active stiffness control of a manipulator in Cartesian coordinates," *Proc. 19th IEEE Conf. Decision and Control*, Albuquerque, N.Mex., December, pp. 95–100.

VIDYASAGAR, M. (1978). *Nonlinear Systems Analysis*, Prentice Hall: Englewood Cliffs, N.J.

WHITNEY, D. E. (1987). "Historical perspectives and state of the art in robot force control," *Int J. Robotics Res.*, Vol. 6, No. 1, pp. 3–14.

WU, J. (1988). "Modeling and control of robotic manipulators," Ph. D. thesis, Clarkson University, Potsdam, N.Y.

YOUNG, K. K. D. (1978). "Controller design for a manipulator using theory of variable structure systems," *IEEE Trans. Syst., Man, Cybern.*, Vol. SMC-8, pp. 101–109.

8

Robot Vision

Modern robotic work cells often contain external sensors that provide the robot controller with information about objects in the environment. Perhaps the most powerful sensor in terms of the volume of data it produces is a robotic vision system. A basic robot vision system has a single stationary camera mounted over the workspace. In some cases, several stationary cameras are used in order to gain multiple perspectives. An unlimited number of perspectives can be achieved with a single mobile camera mounted on the arm. This is an *active* form of sensing in which the robot itself is used to position the sensor on the basis of the results of previous measurements.

Robot vision systems supply valuable information that can be used to automate the manipulation of objects. With the use of robotic vision, the position, orientation, identity, and condition of each part in the scene can be obtained. This high-level information can then be used to plan robot motion such as determining how to grasp a part and how to avoid collisions with obstacles. The subject of robotic vision is broad and deep, and it can easily fill an entire book (Nevatia, 1982; Horn, 1986). We introduce the fundamentals of robotic vision here by restricting our attention to the case of a single stationary overhead camera. Most of the analysis techniques in this chapter also assume that the image from this camera has been thresholded to produce a binary or black-and-white image. The techniques covered include template matching, edge and corner point detection, run length encoding, shape analysis, segmentation, smoothing, perspective transformations, structured illumination, ranging, and camera calibration.

With the effective use of robotic vision, solutions to automated assembly tasks can be made more robust. In particular, clever algorithms which rely on feedback from external sensors (including force and moment sensors) can reduce the sensitiv-

ity to uncertainties in the position, orientation, size, and shape of the parts being assembled. This should make the assembly process more reliable and also reduce the need for expensive part fixturing devices.

8-1 IMAGE REPRESENTATION

To investigate robot vision, we begin by examining how the visual data from the camera is represented. A raw two-dimensional image can be regarded as an analog function $i(x, y)$ which specifies the reflected *light intensity* at each position coordinate (x, y). If the raw image is to be processed with a computer, the analog signal $i(x, y)$ must be converted to an equivalent digital representation. This involves two steps. First the spatial coordinates (x, y) are *sampled* at discrete intervals $(k\Delta x, j\Delta y)$. If there are m samples along the x coordinate and n samples along the y coordinate, then this results in an image with a total of mn picture elements, or *pixels*. In this case we say that the vision system has a *spatial resolution* of $m \times n$ pixels. Often m and n are powers of 2. The storage requirements for an image grow rapidly with the resolution. For example, if the unit of storage is one byte per pixel, then a single frame of a 1024×1024 pixel image uses one megabyte of memory.

The unit of storage for an image will depend on the number of colors or shades of a color that are used to represent the reflected light intensity. The value of the reflected light intensity associated with the pixel in row k and column j will be the *average* light intensity over that picture element. That is:

$$I_a(k, j) \triangleq \frac{\int_0^{\Delta x} \int_0^{\Delta y} i[(k-1)\Delta x + x, (j-1)\Delta y + y]\, dy\, dx}{\Delta x\, \Delta y} \tag{8-1-1}$$

The function $I_a(k, j)$ is nonnegative and takes on values over a *continuum* of intensities, with $I_a(k, j) = 0$ representing no reflected light.

To complete the conversion of $i(x, y)$ to a digital representation, we must also *quantize* the value of $I_a(k, j)$ so it can be represented with a precision of b binary digits, or bits. This results in 2^b possible intensity values, which are commonly referred to as *gray levels*, or shades of gray. We denote the quantized version of $I_a(k, j)$ as simply $I(k, j)$. Thus a *digital image* $I(k, j)$ is an analog image $i(x, y)$ that has been sampled in space to a resolution of $m \times n$ pixels and quantized in intensity to a precision of 2^b gray levels. A pictorial representation of a 5×4 pixel image that has been quantized to eight gray levels is shown in Fig. 8-1. Here the height of each arrow represents the quantized average light intensity over the pixel. Thus Fig. 8-1 represents a bright rectangular 3×2 object centered on a dark 5×4 background.

Exercise 8-1-1: Image Storage. A *binary* image is an image that has only two gray levels, black and white. What is the minimum number of bytes needed to store a 1024×1024 pixel binary image? What are the storage requirements if 16 shades of gray are used?

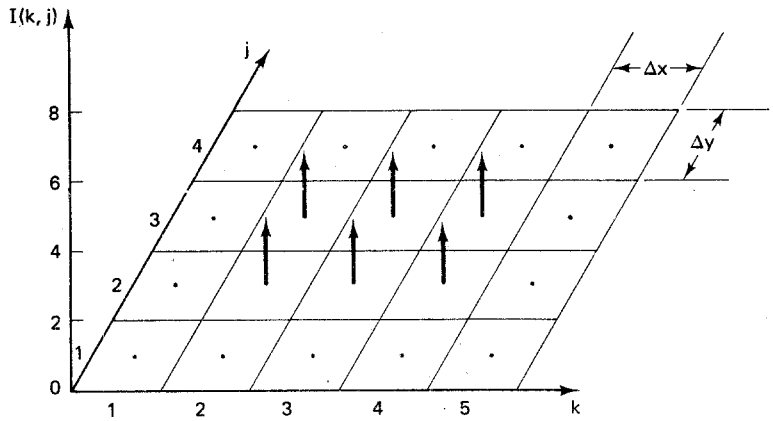


Figure 8-1 A 5 × 4 image with 8 gray levels.

8-2 TEMPLATE MATCHING

One important application of a robot vision system is to recognize whether or not a given part is a member of a particular class of parts. The most straightforward technique for part recognition is to use something called *template matching*. Here representative members, or perhaps characteristic features, of each class of parts are placed in front of the camera, one at a time, and their images are obtained and stored in a *library* of parts. For example, for N classes of parts one could construct the following library:

$$\text{LIB} = \{T_i(k, j): 1 \leq k \leq m0,\ 1 \leq j \leq n0,\ 1 \leq i \leq N\} \qquad (8\text{-}2\text{-}1)$$

The ith representative image, $T_i(k, j)$, is referred to as a *template*, or mask, for class i. Note that template $T_i(k, j)$ is an $m0 \times n0$ image, where $m0 \leq m$ and $n0 \leq n$. An example of a simple 3×3 template of a diamond is shown in Fig. 8-2. Note how the diamond becomes brighter in this case as we proceed from the left to the right.

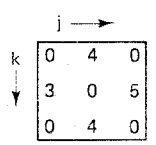


Figure 8-2 A 3 × 3 template of a diamond.

The diamond in Fig. 8-2 might represent an ID marker, or special symbol that is included to identify an object or a particular location on an object. For example, a number of boxes or containers might be placed on a conveyor belt. If the upper left corner of the top of each container has a diamond or some other distinguishing symbol printed on it, then an overhead vision system can be used to locate each container and verify that it is oriented properly.

Given a general $m \times n$ image $I(k, j)$, to determine whether it contains a part

belonging to class i we attempt to find a match with template $T_i(k, j)$. Since template $T_i(k, j)$ is typically smaller than image $I(k, j)$, we search for a match by scanning the image with the template, say, from left to right and top to bottom, as indicated in Fig. 8-3.

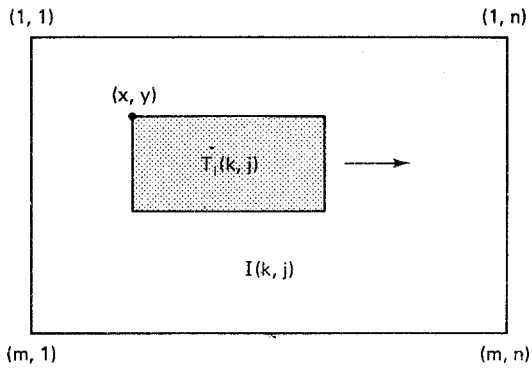


Figure 8-3 Scanning image $I(k, j)$ with template $T_i(k, j)$.

There are numerous ways a match might be measured. Suppose (x, y) represents the *translation* of template $T_i(k, j)$. To avoid overlap of the edges, translations are restricted to $0 \leq x \leq m - m0$ and $0 \leq y \leq n - n0$. If we superimpose $T_i(k, j)$ on $I(k + x, j + y)$, this is equivalent to translating template T_i by an amount (x, y) as shown in Fig. 8-3. For each translation (x, y), the following quantity might be used as a *performance index* for measuring the difference between the image $I(k, j)$ and the translated template $T_i(k, j)$:

$$\rho_i(x, y) \triangleq \sum_{k=1}^{m0} \sum_{j=1}^{n0} |I(k + x, j + y) - T_i(k, j)| \qquad 1 \leq i \leq N \qquad (8\text{-}2\text{-}2)$$

Note that $\rho_i(x, y) \geq 0$, with $\rho_i(x, y) = 0$ if and only if $I(k, j)$ contains $T_i(k, j)$ translated by (x, y). Hence the objective is to search for a translation (x, y) which will minimize the performance index $\rho_i(x, y)$. If an (x, y) is found for which $\rho_i(x, y) = 0$, then an exact match has been detected. The two-dimensional search over all possible translations of the template slows the recognition process down considerably, as there are $m0n0$ subtractions, absolute values, and additions required to evaluate $\rho_i(x, y)$ for each $0 \leq x \leq m - m0$ and $0 \leq y \leq n - n0$. However, if the search is successfully concluded, there is an important additional benefit. Not only is it determined that part i is contained in image $I(k, j)$, but the *location* of the part (x, y) is also identified. A simple implementation of the template matching technique for a library of N parts is summarized in Algorithm 8-2-1.

Algorithm 8-2-1: Template Matching

1. Initialize $i = 1$, $x = 0$, $y = 0$, $\epsilon > 0$, found = true.
2. Compute $\rho_i(x, y)$ using Eq. (8-2-2).
3. If $\rho_i(x, y) \leq \epsilon$, stop.
4. Set $x = x + 1$. If $x \leq m - m0$, go to step 2.
5. Set $x = 0$, $y = y + 1$. If $y \leq n - n0$, go to step 2.

6. Set $x = 0$, $y = 0$, $i = i + 1$. If $i \leq N$, go to step 2.
7. Set found = false.

Note from step 3 of Algorithm 8-2-1 that a match is assumed to have been found if the performance index is less than or equal to some small *threshold* value ϵ. A positive threshold is needed, because it is unrealistic to assume that an exact match will be found. Indeed, there is invariably *noise* present in any raw physical image, particularly an image with many gray levels.

Example 8-2-1: Template Matching

Consider the gray scale image shown in Fig. 8-4. Suppose we try to locate the diamond whose template is in Fig. 8-2 in this 4 × 5 image.

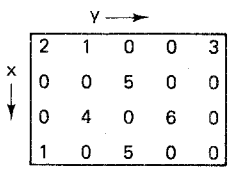


Figure 8-4 A 4 × 5 image.

As the template is 3 × 3 and the image 4 × 5, the possible translations are $0 \leq x \leq 1$ and $0 \leq y \leq 2$. From Eq. (8-2-1), the performance index values corresponding to the translations of the template are:

$$\rho(0, 0) = 8$$
$$\rho(0, 1) = 32$$
$$\rho(0, 2) = 16$$
$$\rho(1, 0) = 30$$
$$\rho(1, 1) = 4$$
$$\rho(1, 2) = 32$$

Thus we see that the best match is $\rho(1, 1) = 4$, but since $\rho(1, 1) > 0$, the match is not perfect. The nonzero performance index is a consequence of the fact that the pattern of shading is somewhat different between the diamond in the template and the diamond in the image.

Although template matching is appealing in view of its conceptual simplicity, a careful analysis reveals that it suffers from a number of inherent practical drawbacks. One limitation is that the two images being compared should have the same *average intensities* $\|T_i\|$ and $\|I_{x,y}\|$. The average intensities in this case are defined as follows:

$$\|T_i\| \triangleq \left[\sum_{k=1}^{m0}\sum_{j=1}^{n0} T_i^2(k, j)\right]^{1/2} \tag{8-2-3a}$$

$$\|I_{x,y}\| \triangleq \left[\sum_{k=1}^{m0}\sum_{j=1}^{n0} I^2(k + x, j + y)\right]^{1/2} \tag{8-2-3b}$$

Two images of the same part often have average intensities that differ. For ex-

ample, the background or the lighting conditions can vary. The sensitivity to the average light intensity can be eliminated if we normalize the performance index. Consider, in particular, the following alternative to the index in Eq. (8-2-2), called the *normalized cross-correlation* function:

$$\sigma_i(x, y) \triangleq \frac{\sum_{k=1}^{m0} \sum_{j=1}^{n0} I(k + x, j + y)T_i(k, j)}{\|I_{x,y}\| \cdot \|T_i\|} \tag{8-2-4}$$

Again, we have $\sigma_i(x, y) \geq 0$. However, in this case we want to find an (x, y) which *maximizes* $\sigma_i(x, y)$. Note from Eq. (8-2-4) that $\sigma_i(x, y)$ reaches a maximum value of unity when there is an exact match with $I(k + x, j + y) = T_i(k, j)$. The normalized cross-correlation index takes on values in the following range:

$$0 \leq \sigma_i(x, y) \leq 1 \tag{8-2-5}$$

Example 8-2-2: Normalized Cross Correlation

Consider again the template in Fig. 8-2 and the image in Fig. 8-4. In this case, the average intensity of the template is $\|T\| = 8.124$. The values for the normalized cross correlation of the translated template with the image are:

$$\sigma(0, 0) = \frac{45}{55.1} = 0.817$$

$$\sigma(0, 1) = \frac{0}{71.7} = 0.000$$

$$\sigma(0, 2) = \frac{39}{70.0} = 0.557$$

$$\sigma(1, 0) = \frac{0}{66.5} = 0.000$$

$$\sigma(1, 1) = \frac{82}{82.1} = 0.999$$

$$\sigma(1, 2) = \frac{0}{75.3} = 0.000$$

The best match again occurs at $\sigma(1, 1) = 0.999$. Here $\sigma(1, 1)$ is quite close to unity, indicating that a good match has been found.

Exercise 8-2-1: Normalized Cross Correlation. The normalized cross-correlation function $\sigma_i(x, y)$ in Eq. (8-2-4) is more traditionally written in the following difference form. Show that this is equivalent to Eq. (8-2-4) by using a change of variables:

$$\sigma_i(x, y) = \frac{\sum_{k=x+1}^{x+m0} \sum_{j=y+1}^{y+n0} I(k, j)T_i(k - x, j - y)}{\|I_{x,y}\| \cdot \|T_i\|}$$

Although the normalized cross correlation in Eq. (8-2-4) effectively addresses the problem of a variation in the average light intensity, there are other, more seri-

ous practical limitations to the template matching technique. They arise from the fact that the method is sensitive to *rotations* of the part, *scale* changes, and *perspective* changes. These variations can, at least in principle, be included in a generalized search strategy. However, doing so is not practical, in view of the extra computational burden. Indeed, the search effort grows exponentially with the number of features or dimensions searched. Another difficulty of the template matching technique is that there may well exist some *variability* within a given class of parts, in which case not every member of the class looks exactly like the representative template or mask. Here techniques have been proposed to stretch, compress, or otherwise deform the template in order to obtain the best match (Widrow, 1973).

Exercise 8-2-2: Template Matching. Find a practical example of an application for which template matching is well suited. The number of classes and the variability within the classes should be reasonably small. *Hint:* What could the post office use?

Exercise 8-2-3: Normalization. Normalize the performance index in Eq. (8-2-2) to remove the effects of the average intensity. Repeat Example 8-2-1, using your normalized performance index.

8-3 POLYHEDRAL OBJECTS

Since robots manipulate three-dimensional objects, it is useful to develop models that efficiently represent such objects. Roberts (1968) initiated studies of three-dimensional scenes by investigating line drawings of polyhedral solids. An example of a polyhedral solid is shown in Fig. 8-5. The tetrahedron is, in fact, the simplest possible three-dimensional object, since it has only four vertices and four identical triangular sides.

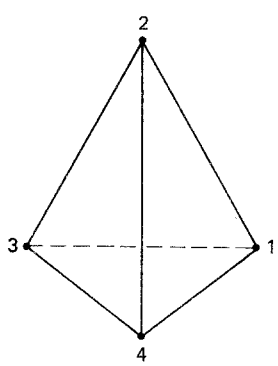


Figure 8-5 A simple polyhedral solid (tetrahedron).

8-3-1 Edge Detection

Suppose the faces of the polyhedral object are smooth, homogeneous, and opaque; and suppose that the lighting is uniform and has been arranged to eliminate shadows. Under these conditions, the reflected light intensity will be constant or will at least

vary continuously over each face of the object. However, the reflected light intensity will have a jump discontinuity at the boundary between adjacent faces. Therefore, at the edges of the polyhedral object, the *gradient* of the light intensity, $\nabla i(x, y)$, will be infinite. Since we are working with a digital image, $I(k, j)$, we must employ a digital approximation of the gradient operator to locate an edge. To develop a numerical approximation to $\nabla i(x, y)$, consider pixel (k, j) and its three neighbors to the south and east as shown in Fig. 8-6.

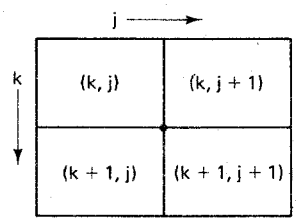


Figure 8-6 Pixels used to approximate the intensity gradient.

Recall that the gradient $\nabla i(x, y)$ is simply the 1×2 Jacobian matrix of partial derivatives of $i(x, y)$ with respect to x and y, respectively. Let $\nabla I_1(k, j)$ and $\nabla I_2(k, j)$ denote numerical approximations to the partial derivatives of $i(x, y)$ with respect to x and y, respectively. Then, computing first differences of the pixels in Fig. 8-6, we have:

$$\nabla I_1(k, j) \triangleq \frac{[I(k+1, j) - I(k, j)] + [I(k+1, j+1) - I(k, j+1)]}{2\Delta x} \tag{8-3-1}$$

$$\nabla I_2(k, j) \triangleq \frac{[I(k, j+1) - I(k, j)] + [I(k+1, j+1) - I(k+1, j)]}{2\Delta y} \tag{8-3-2}$$

Here $\Delta x > 0$ and $\Delta y > 0$ represent the spacing between adjacent pixels in the x and y directions, respectively. Note that $\nabla I_1(k, j)$ is just the average of the first difference in k evaluated at j and $j + 1$. Similarly, $\nabla I_2(k, j)$ is the average of the first difference in j evaluated at k and $k + 1$. The numerical estimate of the gradient in Eqs. (8-3-1) and (8-3-2) is centered, or biased, about the point in the lower right corner of pixel (k, j). To detect an edge, we examine the magnitude or size of the gradient vector. The *magnitude* of the intensity gradient can be computed from Eqs. (8-3-1) and (8-3-2) as:

$$\| \nabla I(k, j) \| = [\nabla I_1^2(k, j) + \nabla I_2^2(k, j)]^{1/2} \tag{8-3-3}$$

An edge is present at pixel (k, j) if the magnitude $\| \nabla I(k, j) \|$ is greater than some *edge threshold* $\epsilon > 0$ which is typically chosen empirically. The edge threshold ϵ should be small enough to pick up all the edges, but it must be large enough to avoid false edges due to noise in the image. The ratio of the pixel spacing in the x direction to the pixel spacing in the y direction is called the *aspect ratio* of the image and is denoted ρ_{xy}:

$$\rho_{xy} \triangleq \frac{\Delta x}{\Delta y} \tag{8-3-4}$$

If the aspect ratio is not equal to 1, then circles in the analog $i(x, y)$ image generate ellipses in the digital $I(k, j)$ image, and conversely. When the aspect ratio

is 1, the magnitude of the intensity gradient in Eq. (8-3-3) simplifies, as can be seen from the following exercise:

Exercise 8-3-1: Roberts' Cross Operator. Suppose the aspect ratio is $\rho_{xy} = 1$ and the pixel spacing is Δx. Show that the norm squared of the approximation to the intensity gradient in Eqs. (8-3-1) and (8-3-2) reduces to the following expression (Roberts, 1968):

$$\|\nabla I(k, j)\|^2 = \frac{[I(k+1, j+1) - I(k, j)]^2 + [I(k, j+1) - I(k+1, j)]^2}{2(\Delta x)^2}$$

The direction of the intensity gradient can also be approximated from $I(k, j)$. In particular, $\nabla I_1(k, j)$ is proportional to $\cos \phi$, and $\nabla I_2(k, j)$ is proportional to $\sin \phi$, where ϕ points in the direction of increasing intensity. Consequently the angle of the numerical approximation to the gradient vector is:

$$\phi(k, j) = \text{atan2}\,[\nabla I_2(k, j), \nabla I_1(k, j)] \tag{8-3-5}$$

Whereas the magnitude information in Eq. (8-3-3) is useful for identifying edge pixels, the angle information in Eq. (8-3-5) can be used to determine which side of the edge corresponds to the object. A simple fixed-threshold algorithm for finding the edges in an image is summarized in Algorithm 8-3-1. Here we assume that the field of view of the original image is sufficiently large that none of the objects in the scene touch the boundary of the $m \times n$ array $I(k, j)$.

Algorithm 8-3-1: Edge Detection

1. Initialize $k = 1, j = 1, \epsilon > 0$.
2. Compute $\|\nabla I(k, j)\|$ using Eq. (8-3-3).
3. If $\|\nabla I(k, j)\| \geq \epsilon$, set $L(k, j) = 1$; otherwise set $L(k, j) = 0$.
4. Set $j = j + 1$. If $j < n$, go to step 2.
5. Set $L(k, n) = 0$.
6. Set $j = 1, k = k + 1$. If $k < m$, go to step 2.
7. For $j = 1$ to n, set $L(m, j) = 0$.

Algorithm 8-3-1 takes an $m \times n$ gray scale image $I(k, j)$ as input and produces an $m \times n$ *binary image* $L(k, j)$ as output, where 1s represent edges. Note that only the first $m - 1$ rows and $n - 1$ columns of $I(k, j)$ are scanned for edge pixels. This is necessary because the magnitude operator in Eq. (8-3-3) requires the use of three neighboring pixels to the east and south. The last row and last column of $L(k, j)$ are filled with 0s.

The key parameter in Algorithm 8-3-1 is the edge threshold ϵ. If ϵ is too large, then only *fragments* of edges will be retrieved. As ϵ decreases, more complete versions of the edges appear in $L(k, j)$, but at some point, as ϵ decreases still further, *false edges* begin to manifest themselves as a result of noise and nonuniform lighting. Algorithm 8-3-1 is a fixed-threshold algorithm that can be used as a step in a higher-level algorithm in which the threshold is adjusted automatically. To develop

a strategy for adjusting ϵ, it is useful to consider the following *edge-pixel distribution* function:

$$N(\epsilon) \triangleq \sum_{k=1}^{m-1} \sum_{j=1}^{n-1} L(k, j) \tag{8-3-6}$$

Here $N(\epsilon)$ represents the total number of edge pixels that have been found in $I(k, j)$. Since the last row and column of $L(k, j)$ are always filled in with 0s, it is clear that $0 \leq N(\epsilon) \leq (m - 1)(n - 1)$. An idealized sketch of an edge-pixel distribution function for a simple, carefully lit scene with 2^b gray levels is shown in Fig. 8-7.

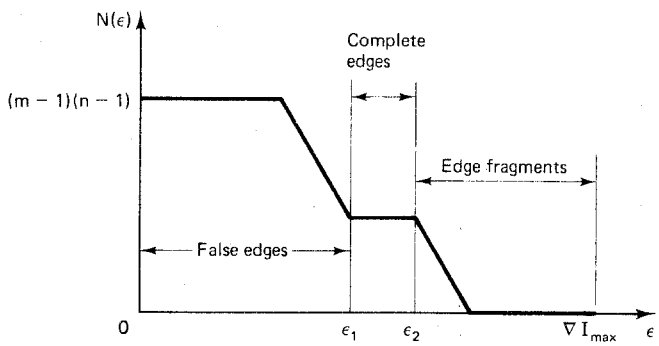


Figure 8-7 Sketch of idealized edge-pixel distribution function.

Notice from Fig. 8-7 that when $\epsilon = 0$, Algorithm 8-3-1 is so sensitive that all of the scanned pixels are regarded as edge pixels. However, as ϵ is increased, the false edges begin to disappear from $L(k, j)$. The edge-pixel distribution function approximately levels off over a small range $[\epsilon_1, \epsilon_2]$, and this represents a reasonably good image $L(k, j)$ in which the edges are more or less complete and few, if any, false edges appear. As the threshold is increased further, Algorithm 8-3-1 becomes too insensitive, and the edges begin to break up into fragments, as can be seen by the fact that the slope of the edge-pixel distribution function again turns negative. Finally, if the threshold is set larger than the maximum possible value of $\| \nabla I(k, j) \|$, no edge pixels will be detected.

It is clear from the idealized case in Fig. 8-7 that the proper range of values for the edge threshold is in the interval $[\epsilon_1, \epsilon_2]$. To locate a point in this interval with an algorithm, it is helpful to work with what might be called an *edge-pixel density* function, that is, the slope, or derivative, of $N(\epsilon)$. Note from Fig. 8-7 that the derivative will have two peaks with a valley between them. The valley corresponds to the interval $[\epsilon_1, \epsilon_2]$.

Exercise 8-3-2: Edge Threshold. Sketch the derivative of the edge-pixel distribution function shown in Fig. 8-7. Write an algorithm for finding a point in the interval $[\epsilon_1, \epsilon_2]$ on the basis of a numerical approximation to the derivative of $N(\epsilon)$.

Once the edge pixels have been determined, they can be fitted with straight lines. The intersections of the straight lines then determine a set of M *vertices*. Since the object is assumed to be a polyhedron, it is completely specified by its vertices. Note that in this case we have taken the original information contained in the mn pixels and *reduced* it to $2M$ integer coordinates which specify the locations of the M vertices of the polyhedral object.

8-3-2 Corner Points

If the image $I(k, j)$ has only two gray levels, it is a *binary*, or black-and-white, image. For binary images, an alternative technique called *corner-point encoding* might be used to extract the vertex pixels directly. Each of the interior pixels has eight adjacent pixels called its *neighbors*. We determine whether a given pixel is a corner point, or vertex, by examining the intensity pattern of its neighbors. This can be done by scanning over the image with a set of 3×3 *corner-point templates*, or masks, which represent all possible types of corners in objects that are at least 2 pixels wide. If $I(k, j)$ is a binary image with 0s representing background pixels and 1s representing foreground pixels, the family of eight corner-point templates shown in Fig. 8-8 can be used.

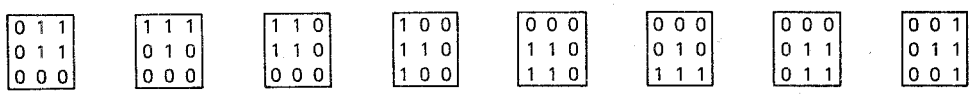


Figure 8-8 Corner-point templates.

Note that the set of corner-point templates is generated by taking the corner pattern appearing in the upper right portion of the first template and rotating it counterclockwise by multiples of $\pi/4$ to generate the remaining seven templates. To search for corner points, we can scan the image with the templates using the normalized cross-correlation function in Eq. (8-2-4). Note from Fig. 8-8 that the center pixel of each template is a foreground pixel. Thus the search for corner points can be made considerably more efficient if we first scan the $(m - 2)(n - 2)$ pixels in the interior of $I(k, j)$ to find *candidate* pixels, pixels with a value of 1. Once a candidate pixel is located, its eight neighbors can then be examined by cross-correlating with the 3×3 templates. If the normalized cross correlation evaluates to 1 for some i, where $1 \leq i \leq 8$, then the process can be terminated, since a corner-point pixel has been identified.

Exercise 8-3-3: Holes in Parts. Suppose an algorithm is available for finding the corner points of a part. Indicate how this same algorithm could be used, without modification, to find the corner points of a hole in the part. *Hint:* The complement of a binary image is a binary image.

Exercise 8-3-4: Whiskers. Suppose we want to locate the end pixels of line segments or curves in the foreground that are only 1 pixel wide. Sketch a set of templates that could be used to identify the end points of these *whiskers*.

8-3-3 Run-Length Encoding

Since high-resolution images contain many pixels, they require a considerable amount of storage space, and they also take a long time to transmit over a communication channel. Images are often *compressed* or encoded to help alleviate these problems. One obvious compression technique for an image with relatively few gray levels is to *pack* several pixels per byte. This reduces the storage of a binary (two-level) image by a factor of 8. However, depending on the size of the image, it may increase the storage and retrieval *time* because the image has to be packed when it is stored and unpacked when it is retrieved.

An alternative compression technique for binary images is to employ a simple encoding scheme called *run-length encoding*. A *run* is a sequence of adjacent pixels that all have the same value, and the *length* of the run is the number of pixels in the sequence. Therefore, instead of storing the values of all of the pixels, we can store the value of the first pixel and the lengths of all of the subsequent runs as we scan the entire image. As long as some convention is followed as to the order in which the pixels are to be scanned, the original image can always be recovered from the encoded image. Depending on the nature of the image, run-length encoding can often reduce the storage space by one or more orders of magnitude.

Example 8-3-1: Run-Length Encoding

Consider the $m \times n$ binary image shown in Fig. 8-9, where $m = 12$ and $n = 18$. Suppose the image is scanned from left to right and top to bottom in the order: $I(1, 1)$, $I(1, 2)$, . . . , $I(m, n)$. From Fig. 8-9, the first pixel is a 0, and the sequence of run lengths is:

$$\Gamma = [0, 44, 6, 10, 9, 7, 4, 3, 4, 5, 5, 5, 3, 7, 5, 1, 5, 8, 9, 12, 3, 16, 1, 44]^T$$

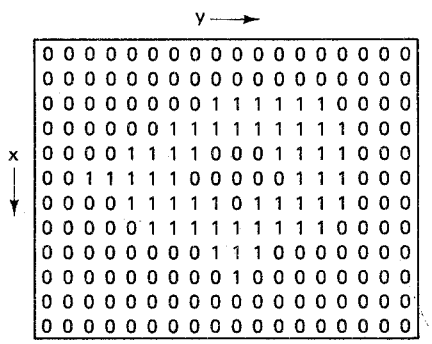


Figure 8-9 Run-length encoding.

Note that since the maximum possible run length is $mn = 216$, each run length can each be stored with one byte. Thus the storage requirement for the encoded image is 24 bytes, including the byte for the color of the starting pixel. Storage of the unpacked binary image requires 216 bytes, and storage of the packed binary image requires 27 bytes. If the part in Fig. 8-7 did not have a hole in the middle, then the run-length code Γ would be even shorter, while the packed binary image would still occupy 27 bytes.

Exercise 8-3-5: Run Length Encoding. Suppose $I(k, j)$ is an $m \times n$ image. For what type of image is the run-length code as short as possible? How many run lengths have to be stored in this case? For what type of image is the run-length code as long as possible? How many run lengths have to be stored in this case?

8-4 SHAPE ANALYSIS

If the parts to be inspected and manipulated are not polyhedral objects, then vertices no longer suffice to describe them. The part boundaries in this case may be curved, and other types of descriptors are needed to characterize the part.

8-4-1 Line Descriptors

There are many examples of objects for which line drawings or edges provide the essential information. For example, the characters in this sentence are all objects which can be characterized by their boundaries. The most straightforward way to represent a line is to use an ordered list of the coordinates of the points. However, rather than store a list of coordinates, it is more efficient to store only the starting coordinates and the *incremental changes* needed to generate successive points on the curve.

Consider, in particular, a pixel p and the labeling scheme for its eight neighbors shown in Fig. 8-10. Here the labeling of the adjacent pixels uses the integers 0 through 7 starting at the east neighbor and proceeding counterclockwise. This technique, called *chain coding,* was introduced by Freeman (1961). Here a *curve* $C(a)$ is represented by a sequence of chain codes $a \in \mathbf{R}^n$, where n is the *length* of the curve in pixels. Chain coding is more efficient than storing the coordinates of the points because it takes only 3 bits per point rather than, for example, 32 bits per point for the coordinates of points in a 512×512 pixel image.

3	2	1
4	p	0
5	6	7

Figure 8-10 Chain codes of neighbors of pixel p.

Another advantage of the chain code representation of a curve is that it is a *relative* representation, whereas a list of coordinates is an *absolute* representation. Two curves of identical shape, but starting at different locations, generate identical chain codes. Thus the chain code of a curve is invariant to *translation.*

Example 8-4-1: Chain Code

Consider the binary image of an object with a curved boundary shown in Fig. 8-11, where 1s represent the object boundary and 0s represent the background. Suppose the chain code of the object boundary is generated by starting at the rightmost pixel and

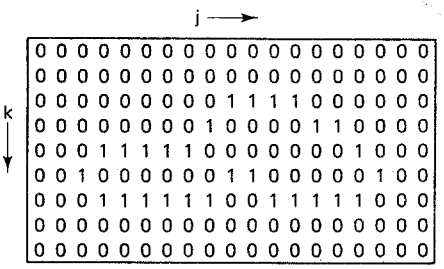


Figure 8-11 Boundary of an object.

traversing counterclockwise. Then the length of the closed curve is $|C(a)| = 28$ pixels, and the chain code of the boundary is:

$$a = [3, 3, 4, 3, 4, 4, 4, 5, 5, 4, 4, 4, 4, 5, 7, 0, 0, 0, 0, 0, 1, 0, 7, 0, 0, 0, 0, 1]^T$$

In some robotic applications, the parts to be manipulated may be only partially visible. The classic example of this is the *bin picking* problem, where parts are dumped into a bin with most parts obscured by other parts. One of the useful features of chain coding is that it can be easily adapted to identify partially occluded boundaries of objects. For example, suppose $C(a)$ represents the boundary of an object with chain code $a \in \mathbf{R}^n$. Next, let $C(b)$ represent a boundary fragment with chain code $b \in \mathbf{R}^m$, where $m < n$. To determine whether the fragment $C(b)$ represents a portion of the curve $C(a)$, we might use the following index to measure the similarity between the two chain codes:

$$\rho_j(a, b) \triangleq \frac{1}{m} \sum_{k=1}^{m} \cos\left[(a_{k+j} - b_k)\frac{\pi}{4}\right] \qquad 0 \leq j \leq n - m \qquad (8\text{-}4\text{-}1)$$

Note from Fig. 8-10 that $a_k \pi/4$ is the angle associated with chain code a_k. Thus the index ρ_j is a normalized sum of the cosines of the difference between the angles of chain codes a_{k+j} and b_k. Since the difference between the angles of any two chain codes will always be a multiple of $\pi/4$, the evaluation of ρ_j for $0 \leq j \leq n - m$ can be performed quite efficiently with a high-speed look-up table scheme, there is no need to compute the cosines directly.

When we evaluate $\rho_j(a, b)$ for j ranging from 0 to $n - m$, this is equivalent to *sliding* the fragment $C(b)$ along the curve $C(a)$. Given the characteristics of the cosine function, the best fit occurs when $\rho_j(a, b)$ is maximum. Since $|\cos \beta| \leq 1$, it is clear that the values of $\rho_j(a, b)$ are restricted to the following interval:

$$-1 \leq \rho_j(a, b) \leq 1 \qquad 0 \leq j \leq n - m \qquad (8\text{-}4\text{-}2)$$

The fragment $C(b)$ represents the *visible portion* of the curve $C(a)$ if $\rho_j(a, b) = 1$, in which case pixel j is the starting point of the fragment $C(b)$ on the curve $C(a)$.

Exercise 8-4-1: Closed Curve. Let $a \in \mathbf{R}^n$ be the chain code of a curve $C(a)$. Find a technique which uses the components of the vector a to determine whether or not $C(a)$ is a *closed curve*. How would you use this technique to determine whether $C(a)$ represents a *simple closed curve*, a closed curve that does not cross over itself?

8-4-2 Area Descriptors

The shape of an object can also be described by an analysis of the points enclosed by the boundary. Whereas boundary descriptors are called *line descriptors*, descriptors based on an analysis of points enclosed by the boundary are called *area descriptors*. Area descriptors tend to be more robust than line descriptors, because a substantial change in the boundary, for example, from smooth to jagged, often results in only a modest change in the enclosed area. Although area descriptors are less sensitive to small changes in the boundary caused by noise, they are more expensive in terms of the required computational effort, because they typically involve processing a larger number of pixels.

To investigate area descriptors, let R represent a *region* in an image $I(k, j)$ associated with some part. We assume that R is a *connected* set, that is, that for each pair of pixels in R there is a path in R which connects the pair. Consequently, R corresponds to a single part, but this part may have one or more holes in it. Initially, we assume that $I(k, j)$ is a binary image with pixels in the lone *foreground* region R having a value of 1 and the remaining pixels representing the *background* having a value of 0. A typical region R is shown in Fig. 8-12.

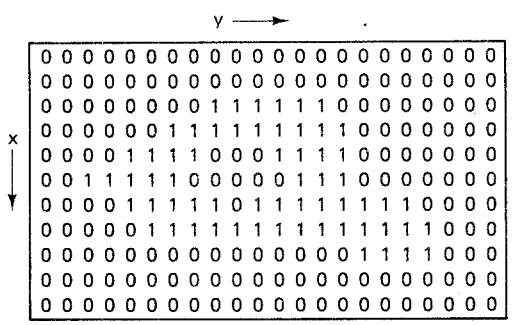


Figure 8-12 A region R in a binary image.

Given a foreground region R, we can compute a sequence of numbers $\{m_{kj}\}$ characterizing its shape called the *moments* of R. We define the moments as follows:

Definition 8-4-1: Moments. Let (x, y) represent the row and column, respectively, of a pixel in a region R. Then the *moments* of the region R are defined:

$$m_{kj} \triangleq \sum_{(x,y)\in R} x^k y^j \qquad k \geq 0, j \geq 0$$

Thus the moments of R are sums of products of integer powers of the row and column numbers of pixels in R. We refer to the sum of the powers, $k + j$, as the *order* of moment m_{kj}. The lower-order moments of R have simple geometric interpretations, as can be seen from the following result:

Proposition 8-4-1: Low-Order Moments. Let $\{m_{kj}\}$ be the moments of a region R. Suppose that the area of R is A pixels and the centroid of R is located at $\{x_c, y_c\}$. Then:

$$A = m_{00}$$

$$x_c = \frac{m_{10}}{m_{00}}$$

$$y_c = \frac{m_{01}}{m_{00}}$$

This follows directly from Def. 8-4-1 and the definitions of the area and centroid of R. In light of Prop. 8-4-1, the zeroth-order moment m_{00} is a measure of the *size* of the region R. Similarly, given the size of the region, the two normalized first-order moments $\{m_{10}/m_{00}, m_{01}/m_{00}\}$ provide us with the *location* of the region R.

Example 8-4-2: Moments

Consider the 8 × 10 binary image shown in Fig. 8-13. Compute the zeroth-, first-, and second-order moments of the foreground region R.

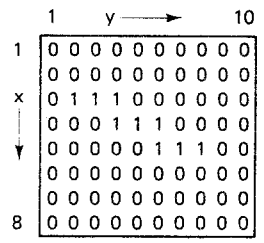


Figure 8-13 A region R in an 8 × 10 binary image.

Solution Applying Def. 8-4-1, the first six moments of R are:

$$m_{00} = 9$$

$$m_{01} = 45 \qquad m_{10} = 36$$

$$m_{02} = 255 \qquad m_{11} = 192 \qquad m_{20} = 150$$

From Prop. 8-4-1, if we normalize the first-order moments with the zeroth-order moment, we have:

$$A = m_{00} = 9$$

$$x_c = \frac{m_{10}}{m_{00}} = 4$$

$$y_c = \frac{m_{01}}{m_{00}} = 5$$

Once the centroid of a region R has been determined, a new set of moments, called the *central moments* of R, can be computed. The central moments tend to have values that are somewhat smaller than the standard moments.

Definition 8-4-2: Central Moments. Let (x_c, y_c) represent the centroid of a region R, and let (x, y) represent the row and column, respectively, of a pixel in R. Then the *central moments* of the region R are defined:

$$\mu_{kj} \triangleq \sum_{(x,y)\in R} (x - x_c)^k (y - y_c)^j \qquad k \geq 0, j \geq 0$$

Note that the central moments $\{\mu_{kj}\}$ can be thought of as the standard moments of R after R has been translated to locate its centroid at the origin. The central moments of R have the property that they are invariant to *translations* of R.

Exercise 8-4-2: Central Moments. Let $\{\mu_{kj}\}$ be the central moments of a region R. Show that the central moments are invariant to translations of R. That is, show that for every region R, $\mu_{10} = 0$ and $\mu_{01} = 0$.

Example 8-4-3: Central Moments

Again, consider the region shown in Fig. 8-13. From Def. 8-4-2 it is clear that $\mu_{00} = 9$, while from Exercise 8-4-2 we have $\mu_{01} = 0$ and $\mu_{01} = 0$. From Example 8-4-2, the coordinates of the centroid of R are $(x_c, y_c) = (4, 5)$. Thus the second-order central moments are:

$$\mu_{02} = 30$$

$$\mu_{11} = 12$$

$$\mu_{20} = 6$$

Just as the zeroth-order and first-order standard moments have simple physical interpretations in terms of the area and centroid, the second-order central moments also have simple interpretations. In particular, the moments μ_{02} and μ_{20} are *moments of inertia* of R corresponding to x and y axes through the centroid, while μ_{11} is called a *product moment*. The central moments can be normalized to produce yet another form of invariance.

Definition 8-4-3: Normalized Central Moments. Let $\{\mu_{kj}\}$ be the central moments of a region R. Then the *normalized central moments* of R are defined:

$$\nu_{kj} \triangleq \frac{\mu_{kj}}{\mu_{00}^{(k+j+2)/2}} \qquad k \geq 0, j \geq 0$$

It is apparent from Def. 8-4-3, that for every region R, the zeroth-order normalized central moment is $\nu_{00} = 1$. Consequently, the normalized central moments $\{\nu_{kj}\}$ are invariant to *scale* changes in R.

Example 8-4-4: Normalized Central Moments

Again, consider the region shown in Fig. 8-13. For every region R, we have $\nu_{00} = 1$, $\nu_{01} = 0$, and $\nu_{10} = 0$. From Example 8-4-2, the size of R is $\mu_{00} = 9$. Thus, from Def. 8-4-3, the second-order normalized central moments are:

$$\nu_{02} = 0.370$$

$$\nu_{11} = 0.148$$

$$\nu_{20} = 0.074$$

8-4-3 Principal Angle

An invariance with respect to *rotations* of R can also be obtained if we make use of something called the *principal angle*. This is a measure of the orientation of R that can be expressed in terms of the second-order central moments, as follows (Nevatia, 1982):

Definition 8-4-4: Principal Angle. Let $\{\mu_{kj}\}$ be the central moments of a region R. Then the *principal angle* of R is defined:

$$\phi \triangleq \tfrac{1}{2} \operatorname{atan2} (2\mu_{11}, \mu_{20} - \mu_{02})$$

To interpret the principal angle physically, consider a line $L(\beta)$ drawn through the centroid of R at an angle of β with respect to the x axis, as shown in Fig. 8-14. The moment of inertia of R about the line $L(\beta)$ will depend on the angle β. The angle at which the moment of inertia is *minimized* is the principal angle $\beta = \phi$. It follows that the principal angle is well defined for an *elongated* object, but it becomes ambiguous when the object approaches a circular shape.

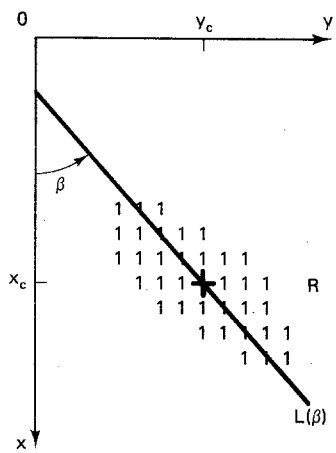


Figure 8-14 The principal angle of a region R.

Example 8-4-5: Principal Angle

Again, consider the small region shown in Fig. 8-13. Using the central moments from Example 8-4-3 and Def. 8-4-4, the principal angle of R is:

$$\phi = \frac{\operatorname{atan2} (24, -24)}{2} = \frac{135}{2} = 67.5 \text{ degrees}$$

Note that this is consistent with the picture in Fig. 8-13, with the angle ϕ being measured from the x axis in a right-handed sense.

Given the principal angle ϕ, we can first translate R by $(-x_c, -y_c)$ and then rotate the translated R by $-\phi$. This will place the centroid at the origin and produce a principal angle of zero. The normalized moments of the resulting region will then be invariant to translations, rotations, and scale changes.

Exercise 8-4-3: Invariant Moments. Find expressions for a set of moments $\{\eta_{kj}\}$ which are invariant to translations, rotations, and scale changes of R.

Example 8-4-6 (Hu, 1962)

Certain combinations of second-order and third-order central moments can be formed which are invariant to rotations and reflections of R. In particular, the following invariant expressions are part of a longer list developed by Hu (1962):

$$M_1 = \mu_{20} + \mu_{02}$$

$$M_2 = (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2$$

$$M_3 = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2$$

$$M_4 = (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2$$

8-5 SEGMENTATION

The process by which items in an image are separated from the background and each other is called *segmentation*. Segmentation involves partitioning the image into connected regions, each region being homogeneous in some sense. For example, small neighborhoods of pixels can be examined for some pattern which is representative of the *texture* of the part. Alternatively, individual pixels can be examined for a particular gray level or perhaps a range of gray levels. The set of all connected pixels with a particular gray level attribute is identified as a *region* and is assigned a unique label.

8-5-1 Thresholding

The simplest way to segment a gray scale image into background and foreground areas is to apply a gray level *threshold* which converts the gray scale image into a binary black-and-white image. Suppose there are b bits used to represent the gray level of each pixel. Hence the intensities, or shades of gray, range from 0 to $2^b - 1$. Let $h(k)$ denote the number of pixels with gray level k. If we plot $h(k)$ versus k, the result is called a *gray scale histogram*, as shown in Fig. 8-15 for the case $b = 6$.

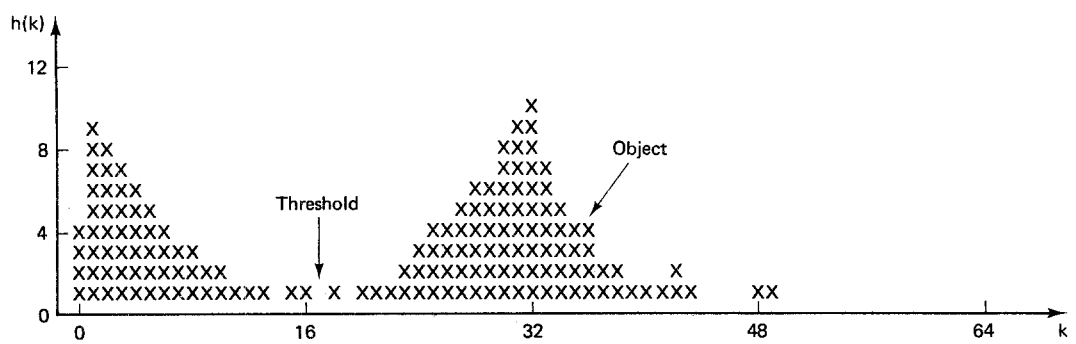


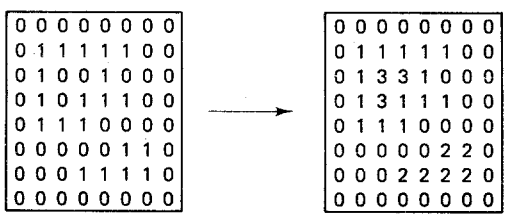
Figure 8-15 Gray scale histogram.

Here a gray level of $k = 0$ represents black, while a gray level of $k = 63$ represents white. With a light object on a dark background, the histogram might look approximately like the *bimodal* distribution in Fig. 8-15. Clearly, the first peak is the background, while the second peak represents the object. There are also some pixels in between which arise from such things as noise, shadows, and nonuniformity in the colors of the object and the background. The area under the histogram is the total number of pixels in the image.

Although there will typically be stray pixels associated with noise and nonuniformity of the image, if the overall histogram exhibits two distinct peaks, then a gray level threshold somewhere between these peaks should segment the image into background and foreground areas. The two areas obtained by gray level thresholding will often not be connected regions. For example, if there are several nonoverlapping foreground objects in the scene, then the foreground area will be disjoint. Similarly, if at least one foreground object has a visible hole in it, then the background area will be disjoint. For robotic applications, we want to assign to each connected region in the foreground a unique *label*. This way, we can apply the shape analysis techniques of Sec. 8-4 to each separate region or part in the scene.

8-5-2 Region Labeling

To develop an algorithm which assigns to each connected foreground region a unique label, suppose we start with a binary image $I(k, j)$ which has been segmented into foreground and background areas. The objective is to produce an updated image in which the connected regions are labeled. As an example of a binary image, consider the 8×8 unlabeled image shown in Fig. 8-16, where 0 and 1 represent background and foreground pixels, respectively.



Unlabeled Labeled

Figure 8-16 Labeling a binary image.

Note that the unlabeled image has two distinct foreground regions, the larger region having a hole in it. In the labeled image, the foreground regions have been assigned the labels 1 and 2. Since the hole is isolated from the other background pixels, it has been assigned a label of 3.

A simple technique for producing a labeled image from a binary image is to use something called *region growing*, or region painting. Here the binary image is scanned for an unlabeled foreground pixel. Each time one is found, it serves as a *seed* for growing a new foreground region. This continues until all of the pixels in the original unlabeled image have been scanned. Consider, in particular, the following algorithm, which assigns unique integer labels to the separate foreground regions in an $m \times n$ binary image.

Algorithm 8-5-1: Region Labeling

1. Initialize $k = 1, j = 1$.
2. If $I(k, j) = 1$, set $I(k, j) = 255$.
3. Set $k = k + 1$. If $k \leq m$, go to step 2.
4. Set $k = 1, j = j + 1$. If $j \leq n$, go to step 2.
5. Initialize $k = 1, j = 1, i = 0$.
6. If $I(k, j) = 255$,
   a. Set $i = i + 1$.
   b. Grow region i from seed (k, j) using Algorithm 8-5-2.
7. Set $k = k + 1$. If $k \leq m$, go to step 6.
8. Set $k = 1, j = j + 1$. If $j \leq n$, go to step 6.

Steps 1 to 4 of Algorithm 8-5-1 preprocess the image by assigning 255, the largest 1-byte integer, to all foreground pixels. In this manner, up to 254 separate foreground regions can be identified. In steps 5 to 8 the image is scanned for unlabeled foreground pixels, pixels with a value of 255. When an unlabeled foreground pixel (k, j) is identified in step 6, it serves as a seed to grow a new foreground region using Algorithm 8-5-2. The output of the region growing procedure is an updated version of the image array $I(k, j)$ in which the pixels in region i have been assigned the label i. A simple region-growing procedure can be implemented if a *stack* data structure is used for temporary storage of pixel addresses (Snyder, 1985). The following is an example of region-growing, or region-painting, algorithm. It is passed a label i, a seed address (k, j), and the image array I. It returns the updated image array I with region i labeled or colored with color i.

Algorithm 8-5-2: Region Growing

1. Set $I(k, j) = i$, push (k, j), push $(0, 0)$.
2. If $j < n$ and $I(k, j + 1) = 255$,
   a. Set $I(k, j + 1) = i$.
   b. Push $(k, j + 1)$.
3. If $k > 1$ and $I(k - 1, j) = 255$,
   a. Set $I(k - 1, j) = i$.
   b. Push $(k - 1, j)$.
4. If $j > 1$ and $I(k, j - 1) = 255$,
   a. Set $I(k, j - 1) = i$.
   b. Push $(k, j - 1)$.
5. If $k < m$ and $I(k + 1, j) = 255$,
   a. Set $I(k + 1, j) = i$.
   b. Push $(k + 1, j)$.
6. Pop (k, j). If $(k, j) \neq (0, 0)$, go to step 2.
7. Pop (k, j), return.

Here step 1 labels the seed, saves the value of the seed address, and puts a (0, 0) marker on the stack. Steps 2 to 5 check the four neighbors of pixel (k, j), starting with the east neighbor and proceeding counterclockwise. Whenever a neighbor is found that is an unlabeled foreground pixel, it is labeled and pushed onto the stack. In step 6, the most recently stored pixel is popped off the stack and its four neighbors are checked. This process is repeated until all connected unlabeled foreground pixels have been found. The address of the seed is then restored in step 7, and control returns to the calling routine. Note that Algorithm 8-5-1 is an *in-place* region-labeling algorithm, in the sense that only the original array $I(k, j)$ is used. Storage for a working copy of $I(k, j)$ is not required. Suppose there are a total of N distinct foreground regions $\{R_i\}$ in the image $I(k, j)$. Once the foreground regions have been identified, area descriptors (moments) can then be computed for each separate foreground region. To determine whether pixel (k, j) belongs to region R_i, it is simply a matter of checking to see whether $I(k, j) = i$.

Exercise 8-5-1: Stack Space. Suppose 4 bytes are used to store a pixel address (k, j). For an $m \times n$ image, what is the maximum amount of memory that could be occupied by the stack?

Exercise 8-5-2: Hole Labeling. Modify Algorithm 8-5-1 so that holes are also assigned unique labels.

The regions $\{R_i\}$ labeled by Algorithm 8-5-1 are referred to as *4-connected* regions, because the search for connected pixels uses only four directions: east, north, west, and south. Thus a diagonal line 1 pixel wide will be labeled as a series of small regions. Of course, the region-growing procedure could be easily modified to search eight directions: east, northeast, north, northwest, west, southwest, south, and southeast. In this case the labeled regions would, in general, be fewer in number but larger in size and would be referred to as *8-connected* regions.

Example 8-5-1: Region Labeling

As an example which illustrates how the region-labeling algorithm works, consider the 8×8 image with two separate foreground regions shown in Fig. 8-17.

In Algorithm 8-5-1, the row index k is incremented faster than the column index j. Thus the image $I(k, j)$ is scanned a column at a time as it works from left to right to identify unlabeled foreground pixels. The first seed is found at address (3, 2), at which point Algorithm 8-5-2 is called with $i = 1$. Note that in Algorithm 8-5-2 the four

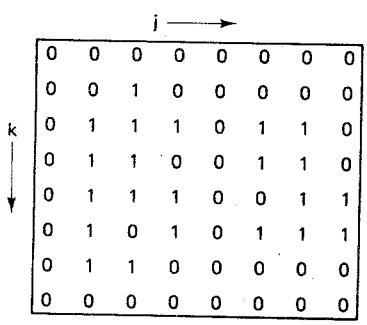


Figure 8-17 An 8×8 image.

neighbors of each unlabeled foreground pixel are searched in the counterclockwise order: east, north, west, south. Neglecting the seed (3, 2) and the marker (0, 0), the contents of the stack at the start of step 6 for each iteration of Algorithm 8-5-2 are shown in Fig. 8-18.

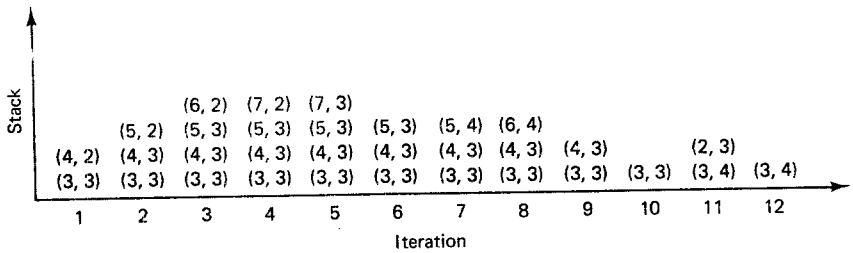


Figure 8-18 Stack contents while growing first region.

At this point control returns to Algorithm 8-5-1 as the growing or coloring of the first region is complete. The partially labeled image array $I(k, j)$ now has the values shown in Fig. 8-19.

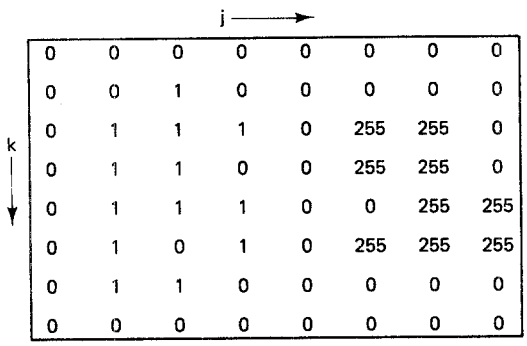


Figure 8-19 Partially labeled image.

The search for unlabeled foreground pixels resumes in Algorithm 8-5-1, starting with pixel (4, 2), where it left off. The next seed is found at address (3, 6), at which point Algorithm 8-5-2 is called with $i = 2$. Again neglecting the seed (3, 6) and the marker (0, 0), the contents of the stack at the start of step 6 for each iteration of Algorithm 8-5-2 are shown in Fig. 8-20.

At this point, control returns to Algorithm 8-5-1, as the growing or coloring of the second region is complete. The labeled image array $I(k, j)$ now has the values shown in Fig. 8-21.

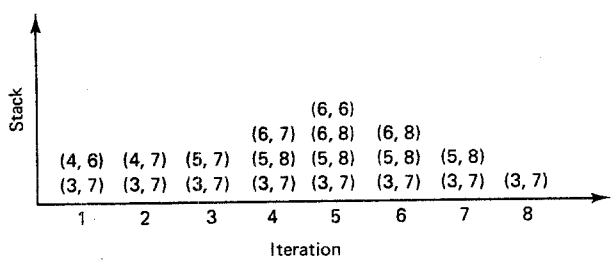


Figure 8-20 Stack contents while growing second region.

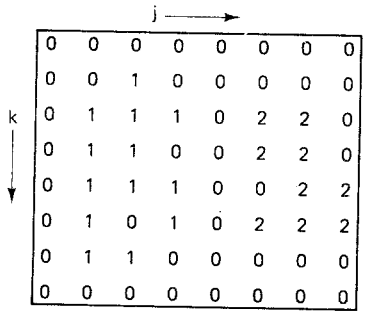


Figure 8-21 Labeled image.

The search for unlabeled foreground pixels resumes in Algorithm 8-5-1, starting with pixel (4, 6), where it left off. In this case no new seed is found, because all foreground pixels have now been labeled. Since $i = 2$ at the termination of Algorithm 8-5-1, there are $N = 2$ separate foreground regions in the image $I(k, j)$.

Exercise 8-5-3: Shape Analysis. Preliminary shape analysis can be performed *concurrently* with region labeling. Indicate how you would modify Algorithm 8-5-2 so that Algorithm 8-5-1 returns the following information upon termination:

$$\{A(i), x_c(i), y_c(i): 1 \leq i \leq N\}$$

Here $A(i)$ is the area of region i and $x_c(i)$ and $y_c(i)$ are the x and y centroids, respectively, of region i.

8-6 ITERATIVE PROCESSING

Binary images typically contain noise due to a variety of sources. Shadows and variations in the lighting can cause the boundaries of regions to appear jagged, with narrow appendages and inlets. If the gray level threshold is set too low, foreground regions tend to merge and small extraneous regions appear in the background area. The gray level threshold can also be set too high, at which point foreground regions split and small holes emerge in the foreground area.

8-6-1 Shrink Operators

Problems of this nature, which are typical in raw visual data, can be minimized by careful control of the background, the lighting, and the selection of a threshold value. In addition to these steps, the raw image $I(k, j)$ can be processed *iteratively* using *local* techniques which assign to each pixel a value that is, in some sense, similar to that of its neighbors. These techniques tend to smooth out rough edges and remove some of the noise. To formulate a family of iterative smoothing operators, consider the following function, $p(k, j)$, based on the eight neighbors of a pixel.

Definition 8-6-1: Pixel Function. Let $I(k, j)$ be an $m \times n$ binary image.

Then the *pixel function* p evaluated at pixel (k, j) is defined:

$$p(k, j) \triangleq \left[\sum_{u=-1}^{1} \sum_{v=-1}^{1} I(k + u, j + v) \right] - I(k, j) \qquad 1 < k < m, 1 < j < n$$

Note that the pixel function $p(k, j)$ simply returns the number of foreground pixels in the immediate neighborhood of pixel (k, j). Thus $p(k, j)$ takes on integer values in the following range:

$$0 \leq p(k, j) \leq 8 \qquad 1 < k < m, 1 < j < n \tag{8-6-1}$$

The pixel function is defined only on the *interior* pixels of an image because the pixels on the border of the image do not have a complete set of neighbors. The utility of the pixel function lies in the fact that it allows us to concisely define a family of operators that act on one image to produce a second image that is, in a sense, smoother. There are two classes of operators that we consider. The first is the shrink operator, which tends to shrink or reduce the size of the foreground regions.

Definition 8-6-2: Shrink Operator. Let $p(k, j)$ be the pixel function, and let $1(\cdot)$ denote the unit step function. Then the *ith shrink operator*, denoted Shrink (i), acting on pixel (k, j) of image $I(k, j)$ is defined:

$$\text{Shrink } (i)\, I(k, j) \triangleq I(k, j) \quad \text{AND} \quad 1(i - 1 - [8 - p(k, j)]) \qquad 0 \leq i \leq 8$$

Note that $8 - p(k, j)$ is the number of 0s surrounding pixel (k, j). Thus the ith shrink operator turns pixel (k, j) into a 0 if pixel (k, j) has at least i neighbors with the value 0. Otherwise, the ith shrink operator leaves pixel (k, j) unchanged. Clearly, the ith shrink operator is *monotonic*, in the sense that the number of foreground pixels in the shrunken image is always less than or equal to the number of foreground pixels in the original image. The shrink operator can be applied *iteratively* to produce a sequence of images $\{I_v(k, j)\}$, where $I_0(k, j) = I(k, j)$ and:

$$I_{v+1}(k, j) = \text{Shrink } (i)\, I_v(k, j) \qquad v \geq 0 \tag{8-6-2}$$

Given the monotonic nature of the shrink operator, the sequence specified in Eq. (8-6-2) will converge in a finite number of iterations, at which point $I_{v+1}(k, j) = I_v(k, j)$. The results of some of the iterative shrink operators are easily predicted. For example, Shrink (0) is not particularly useful, as it turns all interior pixels into background pixels in a single iteration. Shrink (1) accomplishes the same end result if at least one interior background pixel is present in the original image, but it may take many iterations to converge.

The most useful shrink operators are Shrink (i) for $i > 4$. In these cases, the iterative operators tend to preserve more of the original image. For example, Shrink (8) converges in a single iteration to a new image that has all of the isolated foreground pixels removed from the background. In this sense it removes certain noise such as noise arising from a pixel that is always stuck in the 1 state. The iterative Shrink (7) operator removes extraneous foreground regions that are less than or equal to 2 pixels in size. More generally, it removes appendages of arbitrary length that are 1 pixel wide.

Exercise 8-6-1: Shrink Operators. Suppose the operator Shrink (i) is applied iteratively to an $m \times n$ binary image $I(k, j)$. For each value of i, where $0 \leq i \leq 8$, find the maximum number of iterations needed for convergence.

8-6-2 Swell Operators

The shrink operators are useful for eliminating small regions from the background and for removing narrow appendages growing out of regions. However, shrink operators will not fill in small holes or narrow inlets in regions. To achieve these kinds of smoothing operations, we need to use the *dual* of the shrink operator, the swell operator.

Definition 8-6-3: Swell Operator. Let $p(k, j)$ be the pixel function, and let $1(\cdot)$ denote the unit step function. Then the *ith swell operator*, denoted Swell (i), acting on pixel (k, j) of image $I(k, j)$ is defined:

$$\text{Swell } (i)\, I(k, j) \triangleq I(k, j) \quad \text{OR} \quad 1(p(k, j) - i) \qquad 0 \leq i \leq 8$$

Note that the ith swell operator turns pixel (k, j) into a 1 if pixel (k, j) has at least i neighbors with the value 1. Otherwise, the ith swell operator leaves pixel (k, j) unchanged. It follows that the swell operators are *monotonic*, in the sense that the number of foreground pixels in the swollen image is always greater than or equal to the number of foreground pixels in the original image. Like the shrink operator, the swell operator can be applied *iteratively* to produce a sequence of images $\{I_v(k, j)\}$, where $I_0(k, j) = I(k, j)$ and:

$$I_{v+1}(k, j) = \text{Swell } (i)\, I_v(k, j) \qquad v \geq 0 \qquad (8\text{-}6\text{-}3)$$

Given the monotonic nature of the swell operator, the sequence specified in Eq. (8-6-3) will converge in a finite number of iterations, at which point $I_{v+1}(k, j) = I_v(k, j)$. As with the shrink operators, the results of some of the iterative swell operators are easily predicted. For example, Swell (0) turns all interior pixels into foreground pixels in a single iteration. Swell (1) accomplishes the same end result if at least one interior foreground pixel is present in the original image, but it may take many iterations to converge.

The most useful swell operators are Swell (i) for $i > 4$. In these cases, the iterative swell operators preserve more of the original image. For example, Swell (8) converges in a single iteration to a new image that has all of the single-pixel holes in foreground regions filled in. In this sense it removes certain noise such as noise arising from a pixel that is always stuck in the 0 state. The iterative Swell (7) operator fills in holes that are less than or equal to 2 pixels in size. It also fills in inlets of arbitrary length that are 1 pixel wide.

Exercise 8-6-2: Swell Operators. Show that the swell operators are redundant, in the sense that Swell (i) can be expressed in terms of shrink operators. *Hint:* Use De Morgan's laws.

Whereas the shrink operators remove small regions and eliminate narrow ap-

pendages, the swell operators fill in small holes and narrow inlets. The small regions in the background and small holes in the foreground are referred to as *salt-and-pepper noise*. To remove both the salt and the pepper, and to smooth out jagged edges in general, it is often effective to use both shrink and swell operators. For example, one can first shrink an image until it converges and then swell the shrunken image. This tends to remove small holes, small regions, narrow inlets, and narrow appendages. Alternatively, one can first swell an image until it converges and then shrink the swollen image.

Example 8-6-1: Image Smoothing

As an example of an application of the shrink and swell operators, consider the 64 × 128 image shown in Fig. 8-22. Note that there are three main foreground regions with jagged edges and a small amount of salt-and-pepper noise. The Shrink (5) operator was applied once, and this was followed by one application of the Swell (5) operator, with the result shown in Fig. 8-23. A comparison with Fig. 8-22 reveals that the salt-and-pepper noise has been removed and the edges have been made smoother.

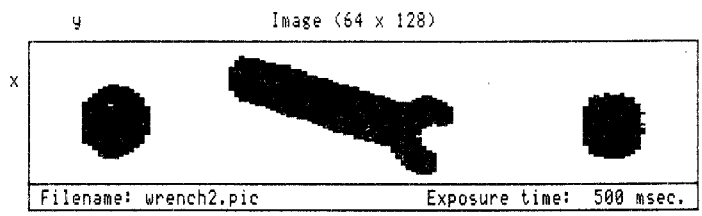


Figure 8-22 A raw 64 × 128 binary image.

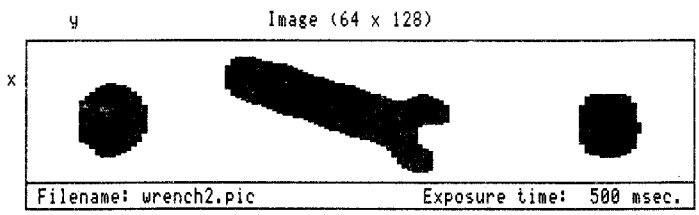


Figure 8-23 The processed 64 × 128 binary image.

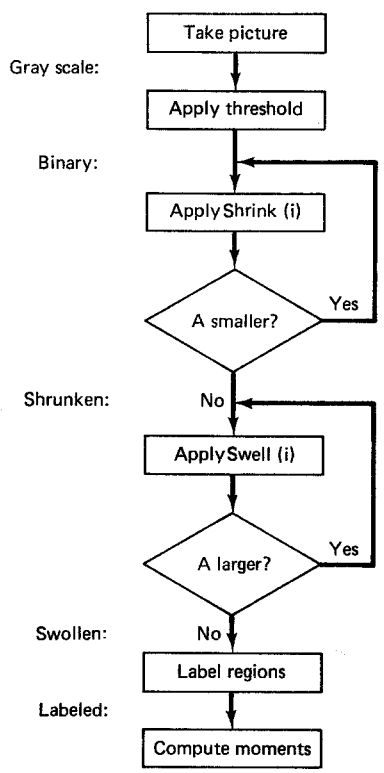
Given a raw gray scale image, we now have a series of processing steps that we can apply in order to extract information useful for planning the motion of a robotic arm. Algorithm 8-6-1 summarizes these image analysis operations.

First the gray scale image is thresholded, with the threshold value chosen through an analysis of the gray scale histogram. This produces a binary image segmented into foreground and background areas. Next, an iterative Shrink (i) operator is applied for some $i > 4$. A simple test for convergence examines the total foreground area after each application of Shrink (i). If A_v denotes the total foreground area after the vth iteration of the Shrink (i) operator, then:

$$A_v \triangleq \sum_{k=1}^{m} \sum_{j=1}^{n} I_v(k, j) \tag{8-6-4}$$

In general, $A_{v+1} \leq A_v$, because the foreground regions are shrinking. As soon as $A_{v+1} = A_v$, the shrinking operation is complete. Next, an iterative Swell (i) operator is applied for some $i > 4$. In this case $A_{v+1} \geq A_v$ in general, because the foreground regions are growing. Once $A_{v+1} = A_v$, the swelling process has converged and the image smoothing phase is complete. The distinct foreground regions in the smoothed binary image are then assigned unique labels using Algorithm 8-5-1. On the basis of these labels, the moments $\{m_{00}, m_{10}, m_{01}, \mu_{20}, \mu_{11}, \mu_{02}\}$ are computed for each region. From these moments, the area A, centroid (x_c, y_c), and principal angle ϕ of each foreground region or part can be determined. The centroid and the principal angle determine the position and orientation of the part, while the area is helpful in distinguishing one part from another. Other characteristics such as the perimeter, the length, the width, or a measure of elongation such as the length-to-width ratio can also be used to discriminate between parts.

Algorithm 8-6-1: Image Analysis



8-6-3 Euler Number

Area, perimeter, length, and width are *geometric* characteristics. It is also possible to classify parts by *topological* characteristics. Topological characteristics depend on the notion of neighborhood and whether or not a given pixel is *connected* to another

pixel. An important topological characteristic which can be applied to an entire image or to a region within an image is the Euler number.

Definition 8-6-4: Euler Number. The *Euler number* of an image is the number of parts minus the number of holes.

Here a *part* is a connected foreground region, while a *hole* is an isolated background region enclosed by a part. If the image has only one part, then the Euler number of the part is 1 minus the number of holes in the part. For example, the letters A, B, and C have Euler numbers of 0, -1, and 1, respectively.

Exercise 8-6-3: Euler Numbers. Find the Euler numbers of the digits 0, 1, . . . , 9, the phrase *Euler number*, and a stencil containing the uppercase letters A, B, . . . , Z.

When Def. 8-6-4 is applied to a discrete image $I(k, j)$, there can be some ambiguity as to whether or not a set of background pixels constitutes a hole. Consider, for example, the image shown in Fig. 8-24. If the region-labeling algorithm (Algorithm 8-5-1) is applied here, the labeled image will have four distinct foreground regions. Each of these regions is a solid 2×3 rectangle that does not have a hole in it. Consequently, it appears that the Euler number of the image in Fig. 8-24 is 4. The ambiguity arises from the fact that if we now complement the image in Fig. 8-24 and then apply Algorithm 8-5-1 to label the background regions, there are two distinct background regions, thus suggesting that the Euler number should instead be 3. Here the 2×3 region of zeros in the center of the image is *isolated* from the rest of the background. However, it is not enclosed by any of the foreground regions; rather, it is encircled by the *arrangement* of foreground regions. The easiest way to resolve this ambiguity is to reconsider our notion of connectedness.

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0	0
0	0	1	1	1	0	0	0	1	1	1	0	0	0
0	0	1	1	1	0	0	0	1	1	1	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-24 Computing the Euler number of an Image.

Definition 8-6-5: Connectedness. A pixel is *4-connected* to its neighbors if and only if at least one of the four pixels to the east, north, west, or south has the same value. A pixel is *8-connected* to its neighbors if and only if at least one of its eight neighbors has the same value.

Note that 4-connectedness is a stronger criterion than 8-connectedness, because every 4-connected region is also 8-connected but the converse does not hold. To resolve the ambiguity posed by the image in Fig. 8-24, we use two different

types of connectedness, one for the foreground and another for the background:

Foreground regions: 4-connected
Background regions: 8-connected

Thus foreground regions must be 4-connected as in Algorithm 8-5-1, but for background regions we use the weaker criterion; they can be 8-connected. Using this convention, the 2 × 3 region of zeros encircled by the four foreground regions in Fig. 8-24 is, in fact, connected to the rest of the background. Hence the Euler number of the image is 4.

An alternative way to resolve this difficulty is to use a different *tesselation* scheme for the image in which every other row is shifted by half a pixel and the pixels have six sides, as shown in Fig. 8-25. Within this hexagonal grid, each pixel has six neighbors, and in this case the notion of *6-connectedness* can be used for *both* the foreground regions and the background regions (Horn, 1986).

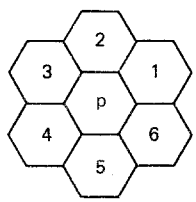


Figure 8-25 A hexagonal tesselation scheme.

It is clear that the shrink and swell operators used for smoothing the image do *not* preserve the Euler number of the image. The shrink operators eliminate small foreground regions from the background; hence the Euler number of a shrunken image is less than or equal to the Euler number of the original image. Similarly, the swell operators fill in small holes in regions; hence the Euler number of a swollen image is greater than or equal to the Euler number of the original image. The shrink and swell operators can be modified to preserve the Euler number of an image (Horn, 1986). An iterative operator which shrinks the image as much as possible while preserving the Euler number is called a *skeleton operator*. When an iterative skeleton operator is applied to a simple region, it reduces it to a point, whereas if it is applied to a region with a single hole, the final result is a thin ring. The dual of a skeleton operator might be called a *bulk operator*. A bulk operator swells an image as much as possible while preserving the Euler number.

Since the skeleton and bulk operators preserve the Euler number of the image, they are not effective in removing salt-and-pepper noise from an image. However, the skeleton and bulk operators can be used effectively in sequence to highlight *defects* in parts. As an example, consider an image of electrical traces on a printed circuit board. There are two common types of manufacturing defects or faults that occur: a *break* fault, or gap in a trace; and a *bridge* fault, or short circuit between adjacent traces. Both of these faults can be hard to detect by direct inspection of the original image because they can be caused by an extremely narrow crack or line.

To highlight these faults, one can first shrink the image with several iterations of a skeleton operator and then swell the image with several iterations of a bulk op-

erator. If a break fault is present in a trace, the break or gap will grow with each application of the skeleton operator and therefore become more noticeable, as shown in Fig. 8-26. Alternatively, if a bridge fault is present between two traces, then the bridge will be preserved by the skeleton operator. Consequently, when the image is again fleshed out by the bulk operator, the bridge will widen and become more noticeable, as shown in Fig. 8-27.

Figure 8-26 Highlighting a break fault with a skeleton operator.

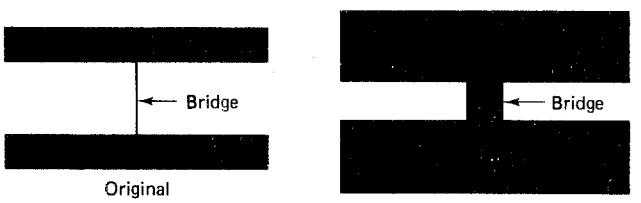


Figure 8-27 Highlighting a bridge fault with a bulk operator.

8-7 PERSPECTIVE TRANSFORMATIONS

A robotic vision system can be used to identify the coordinates of certain points on a part, say, the vertices. Suppose we want to determine whether these points correspond to a model stored in a library. At first glance it might appear that we simply use some direct measure of the difference between the two sets of points. The problem with this approach is that the part may have been translated, rotated, or scaled relative to the model, in which case it is viewed by the camera from a different perspective.

8-7-1 Perspective Transformation

To investigate the effects of perspective, consider the case when a camera is stationed overhead and a part is placed directly below the camera as shown in Fig. 8-28. Note that the origin of the *camera coordinate frame*, $C = \{x^c, y^c, z^c\}$, is located in the *image plane* of the camera and the z^c axis of the camera frame is aligned with the optical axis of the lens. The distance between the lens and the image plane is called the *effective focal distance* f. We assume that the *depth of field* of the camera is sufficiently large that all parts resting on the work surface are in focus.

Consider a point p on a part. The coordinates of p with respect to the camera coordinate frame C are denoted p^{camera}. The coordinates of the image of p, again with respect to the camera coordinate frame C, are denoted p^{image}. The components of p^{image} can be computed in terms of the components of p^{camera} using trigonometry.

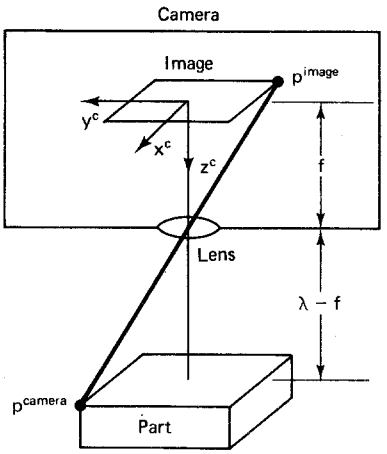


Figure 8-28 A part aligned with the camera.

In particular, from Fig. 8-28, using similar triangles inside and outside the camera, we have:

$$-\frac{p_1^{image}}{f} = \frac{p_1^{camera}}{p_3^{camera} - f} \tag{8-7-1}$$

$$-\frac{p_2^{image}}{f} = \frac{p_2^{camera}}{p_3^{camera} - f} \tag{8-7-2}$$

$$p_3^{image} = 0 \tag{8-7-3}$$

Note that $p_3^{image} = 0$ because the origin of the camera coordinate frame is in the image plane. Thus we have a two-dimensional image of a three-dimensional part, as seen from above. The coordinates of the image of p can be obtained from the coordinates of p by recasting Eqs. (8-7-1) to (8-7-3) in vector form as:

$$p^{image} = \begin{bmatrix} -\dfrac{fp_1^{camera}}{p_3^{camera} - f} \\ -\dfrac{fp_2^{camera}}{p_3^{camera} - f} \\ 0 \end{bmatrix} \tag{8-7-4}$$

The resulting transformation from p^{camera} to p^{image} is called the *perspective transformation*. Note that the perspective transformation is a *nonlinear* mapping from $\mathbf{R}^3$ into $\mathbf{R}^3$ that involves division by p_3^{camera}. The denominators of the components p_1^{image} and p_2^{image} are always positive, since the part is outside the camera. The negative signs in the numerators indicate that the camera *inverts* the image. Of course, when the image is processed digitally and displayed on a computer monitor, it can be reinverted to restore the original orientation. This postprocessing of the image has the effect of putting the image plane in front of the camera, which is equivalent to making the effective focal distance f negative.

Since the perspective transformation is a nonlinear mapping from $\mathbf{R}^3$ into $\mathbf{R}^3$, it cannot be represented by a 3×3 matrix. However, we can represent it with a ma-

trix operator in a higher-dimensional space, the space of four-dimensional homogeneous coordinates. Recall from Chap. 2 that if p is a point in $\mathbf{R}^3$ and F is an orthonormal coordinate frame for $\mathbf{R}^3$, then the *homogeneous coordinates* of p with respect to F are denoted $[p]^F$ and defined:

$$[p]^F = [\sigma p_1, \sigma p_2, \sigma p_3, \sigma]^T \qquad \sigma \neq 0 \tag{8-7-5}$$

Here the augmented fourth component, σ, is a nonzero *scale factor*. In Chap. 2 we used $\sigma = 1$ which is standard scaling. We recover the physical coordinates $p \in \mathbf{R}^3$ from the homogeneous coordinates $[p]^F \in \mathbf{R}^4$ by normalizing with the scale factor as follows:

$$p_k = \frac{[p]_k^F}{\sigma} \qquad 1 \leq k \leq 3 \tag{8-7-6}$$

Clearly, the homogeneous coordinates of a point are not unique, because *any* scale factor $\sigma \neq 0$ can be used. We exploit this particular feature to establish the following result, which provides us with a 4×4 matrix representation of the perspective transformation:

Proposition 8-7-1: Perspective Transformation. Let $p^{\text{camera}} \in \mathbf{R}^4$ be the homogeneous coordinates of a point with respect to a camera frame C, and let $p^{\text{image}} \in \mathbf{R}^4$ be the homogeneous coordinates of the image of the point, again with respect to the camera frame C, as shown in Fig. 8-28. Then $p^{\text{image}} = T_{\text{image}}^{\text{camera}} p^{\text{camera}}$, where:

$$T_{\text{image}}^{\text{camera}} = \left[\begin{array}{ccc:c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hdashline 0 & 0 & -1/f & 1 \end{array}\right]$$

Proof. Since we have a candidate for the perspective transformation, it is simply a matter of verifying that it works. Using the general properties of homogeneous coordinates and Eq. (8-7-4), we have:

$$\begin{aligned} T_{\text{image}}^{\text{camera}} p^{\text{camera}} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix} p^{\text{camera}} \\ &= \left[p_1^{\text{camera}}, p_2^{\text{camera}}, 0, 1 - \frac{p_3^{\text{camera}}}{f} \right]^T \\ &= \left[p_1^{\text{camera}}, p_2^{\text{camera}}, 0, \frac{f - p_3^{\text{camera}}}{f} \right]^T \\ &= \left[\frac{f p_1^{\text{camera}}}{f - p_3^{\text{camera}}}, \frac{f p_2^{\text{camera}}}{f - p_3^{\text{camera}}}, 0, 1 \right]^T \\ &= p^{\text{image}} \end{aligned}$$

We refer to $T_{\text{image}}^{\text{camera}}$ as the *perspective transformation matrix*. Note that it maps camera coordinates into image coordinates. That is, it allows us to determine the coordinates of the image of a point with respect to the camera frame, given the coordinates of the point with respect to the camera frame.

8-7-2 Inverse Perspective Transformation

Although the perspective transformation $T_{\text{image}}^{\text{camera}}$ is useful for analysis purposes, it is really the *inverse perspective transformation* $T_{\text{camera}}^{\text{image}}$ that is the key to using vision to plan the motion of a robotic arm. Consider, for example, the robotic workstation with overhead camera shown in Fig. 8-29. The objective is to find $T_{\text{base}}^{\text{part}}$, the position and orientation of the part frame $P = \{x^p, y^p, z^p\}$ with respect to the robot base frame $B = \{x^b, y^b, z^b\}$. Once this is known, a value for the arm matrix $T_{\text{base}}^{\text{tool}}$ needed to reach down and manipulate the part can be determined.

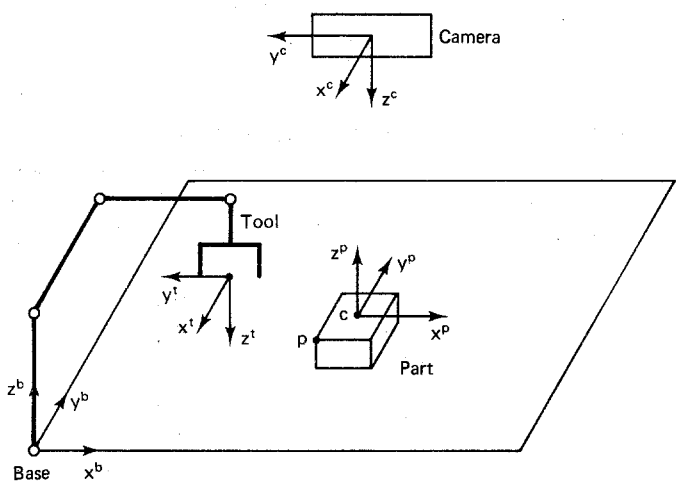


Figure 8-29 A robotic workstation.

If the camera is stationary, then $T_{\text{base}}^{\text{camera}}$, the position and orientation of the camera frame C relative to the robot base frame B, is fixed and can be measured. In order to determine $T_{\text{base}}^{\text{part}}$, we need to know $T_{\text{camera}}^{\text{part}}$. Indeed, once the position and orientation of the part frame relative to the camera frame is determined, we then have:

$$T_{\text{base}}^{\text{part}} = T_{\text{base}}^{\text{camera}} T_{\text{camera}}^{\text{part}} \tag{8-7-7}$$

The position and orientation of the part relative to the camera must be obtained from the images of several points on the part. That is, we must find a transformation from p^{image} to p^{camera} for various points p on the part. The transformation which maps the coordinates of the image of a point into the coordinates of the point is the inverse perspective transformation, $T_{\text{camera}}^{\text{image}}$. Since we already have an expression for the perspective transformation matrix in Prop. 8-7-1, it would appear that it is simply a matter of computing the matrix inverse. Unfortunately, inspection of $T_{\text{image}}^{\text{camera}}$ in Prop. 8-7-1 reveals that the perspective transformation matrix is *singular*

and therefore does not have an inverse! On reflection, this is to be expected, because the perspective transformation compresses three-dimensional objects into two-dimensional images of objects. Depth information is destroyed in the process, with smaller objects closer to the camera having images identical to larger objects farther from the camera.

Since we are attempting to go from a two-dimensional image of a point to the three-dimensional coordinates of the point, it is clear that supplementary information is needed to recover the location of the point. Suppose, for the moment, that the third coordinate p_3^{camera} is known *a priori*. This might be the case, for example, if the part is resting on the work surface and the height of the part is known. Alternatively, the depth information might be obtained from a separate direct measurement of p_3^{camera}. There are several *ranging* techniques that might be used for this purpose, including ultrasonic ranging, infrared ranging, triangulation, and stereoscopic vision. Whatever the method used to determine p_3^{camera}, once this depth information is known it can then be used to augment the image coordinates p^{image} in the following manner, where i^3 denotes the third column of the 4×4 identity matrix:

$$p^{\text{aug}}(\lambda) \triangleq p^{\text{image}} - \left(\frac{\lambda}{\lambda - f}\right) i^3 \tag{8-7-8a}$$

$$\lambda \triangleq p_3^{\text{camera}} \tag{8-7-8b}$$

For convenience, we use the notation $\lambda = p_3^{\text{camera}}$. We refer to the vector $p^{\text{aug}}(\lambda)$ as the *augmented image coordinates* of the point p. Note from Eqs. (8-7-4) and (8-7-8) that the expression for $p_3^{\text{aug}}(\lambda)$ is generally similar in form to the expressions for p_1^{image} and p_2^{image}. In effect, we have *lifted* the image off the image plane using depth information either known a priori or obtained from a separate range measurement. Given the depth information embodied in λ, we see from Eq. (8-7-8) that the transformation from two-dimensional image coordinates to three-dimensional augmented image coordinates can be represented with the following homogeneous translation matrix:

$$T_{\text{aug}}^{\text{image}}(\lambda) = \text{Tran}\left(\frac{\lambda i^3}{f - \lambda}\right) \tag{8-7-9}$$

The augmented image coordinates $p^{\text{aug}}(\lambda)$ contain sufficient information to be transformed directly into camera coordinates p^{camera}. Thus image coordinates can be transformed into augmented image coordinates, and then augmented image coordinates can be transformed into camera coordinates. The composition of these two operations generates the following formulation of the inverse perspective transformation:

Proposition 8-7-2: Inverse Perspective. Let $p^{\text{camera}} \in \mathbf{R}^4$ be the homogeneous coordinates of a point with respect to a camera frame C, and let $p^{\text{image}} \in \mathbf{R}^4$ be the homogeneous coordinates of the image of the point, again with respect to the camera frame C, as shown in Fig. 8-28. If $\lambda = p_3^{\text{camera}}$ is available from a separate range measurement, then $p^{\text{camera}} = T_{\text{camera}}^{\text{image}}(\lambda) p^{\text{image}}$, where:

$$T_{\text{camera}}^{\text{image}}(\lambda) = \left[\begin{array}{ccc:c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & f & \dfrac{f\lambda}{f-\lambda} \\ \hdashline 0 & 0 & 1 & \dfrac{f}{f-\lambda} \end{array}\right]$$

Proof. Using Eq. (8-7-4) and the general properties of homogeneous coordinates:

$$\begin{aligned} T_{\text{camera}}^{\text{image}}(\lambda)p^{\text{image}} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & f & \dfrac{f\lambda}{f-\lambda} \\ 0 & 0 & 1 & \dfrac{f}{f-\lambda} \end{bmatrix} \begin{bmatrix} p_1^{\text{image}} \\ p_2^{\text{image}} \\ 0 \\ 1 \end{bmatrix} \\ &= \left[p_1^{\text{image}},\ p_2^{\text{image}},\ \frac{f\lambda}{f-\lambda},\ \frac{f}{f-\lambda}\right]^T \\ &= \left[\frac{(f-\lambda)p_1^{\text{image}}}{f},\ \frac{(f-\lambda)p_2^{\text{image}}}{f},\ \lambda,\ 1\right]^T \\ &= [p_1^{\text{camera}},\ p_2^{\text{camera}},\ p_3^{\text{camera}},\ 1]^T \\ &= p^{\text{camera}} \end{aligned}$$

The following example is an illustration of the use of Prop. 8-7-2 to compute the coordinates of a point with respect to a robot, given the coordinates of the image of the point with respect to a camera.

Example 8-7-1: Inverse Perspective

Consider the robotic workstation shown in Fig. 8-29. Suppose that the coordinates of the origin of the camera frame $C = \{x^c, y^c, z^c\}$ with respect to the robot base frame $B = \{x^b, y^b, z^b\}$ are determined from measurements to be $[30, 40, 54]^T$ cm. The effective focal distance of the camera is $f = 1$ cm, and the height of the part is $h = 3$ cm. The problem is to find p^{base}, the coordinates of the point p with respect to the robot base frame B, if the coordinates of the image of p with respect to the camera frame C are:

$$p^{\text{image}} = [-0.12, -0.06, 0, 1]^T \quad \text{cm}$$

Solution First we find the necessary depth parameter λ. As the part is 3 cm high and the camera frame is 54 cm above the work surface, it follows that $\lambda = 51$ cm. Using $f = 1$ cm and Prop. 8-7-2, we then have:

$$\begin{aligned} p^{\text{camera}} &= T_{\text{camera}}^{\text{image}}(51)p^{\text{image}}, \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1.02 \\ 0 & 0 & 1 & -0.02 \end{bmatrix} \begin{bmatrix} -0.12 \\ -0.06 \\ 0 \\ 1 \end{bmatrix} \end{aligned}$$

$$= [-0.12, -0.06, -1.02, -0.02]^T$$

$$= [6, 3, 51, 1]^T \text{ cm}$$

From p^{camera} we obtain p^{base}, using the camera-to-base coordinate transformation matrix $T_{\text{base}}^{\text{camera}}$. The coordinates of the origin of the camera frame with respect to the base frame are $[30, 40, 54]^T$ cm. This represents the translation part of $T_{\text{base}}^{\text{camera}}$. We obtain the rotation part from inspection of the orientations of $\{x^c, y^c, z^c\}$ relative to B in Fig. 8-29. This yields:

$$p^{\text{base}} = T_{\text{base}}^{\text{camera}} p^{\text{camera}}$$

$$= \begin{bmatrix} 0 & -1 & 0 & 30 \\ -1 & 0 & 0 & 40 \\ 0 & 0 & -1 & 54 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 3 \\ 51 \\ 1 \end{bmatrix}$$

$$= [27, 34, 3, 1]^T \text{ cm}$$

Note that this is consistent with the picture in Fig. 8-29, given the position of the camera, the camera coordinates of p, and the height h of the part.

Exercise 8-7-1: Pick Position. The origin c of the part coordinate frame in Fig. 8-29 is located at the centroid of the part. Suppose the coordinates of the image of c with respect to the camera frame are:

$$c^{\text{image}} = [-0.04, -0.02, 0, 1]^T \text{ cm}$$

Given the parameters specified in Example 8-7-1 and the results of Example 8-7-1, what are the dimensions of the part? What is an appropriate value for the arm matrix $T_{\text{base}}^{\text{tool}}$ to command the robot to reach down and pick up the part?

8-7-3 Pixel Coordinates

Recall from Prop. 8-7-2 that to use the inverse perspective transformation, we need the coordinates of the image of the point p with respect to the camera frame C. These coordinates, denoted p^{image}, are not directly available from the image. Instead, the row and column numbers (k, j) of a pixel in the $m \times n$ sensing array associated with p are available from the vision system. The pair (k, j) represent the coordinates of the image of p with respect to a *pixel coordinate frame* attached to the sensing array.

Consider, for example, the $m \times n$ sensing array shown in Fig. 8-30, where $m = 8$ and $n = 16$. Here the origin of the pixel coordinate frame is in the upper left corner of the sensing array. The x^{pix}, or k, axis points down, and x coordinates are expressed in units of integer multiples of the pixel height Δx. The y^{pix}, or j, axis points to the right, and y coordinates are expressed in units of integer multiples of the pixel width Δy. The z^{pix} axis completes the right-handed pixel frame $P = \{x^{\text{pix}}, y^{\text{pix}}, z^{\text{pix}}\}$ and therefore points out of the page. Note that the camera coordinate frame $C = \{x^c, y^c, z^c\}$ is in the center of the sensing array and is oriented in a manner consistent with Fig. 8-28.

To determine the pixel-to-image coordinate transformation matrix $T_{\text{image}}^{\text{pixel}}$, we

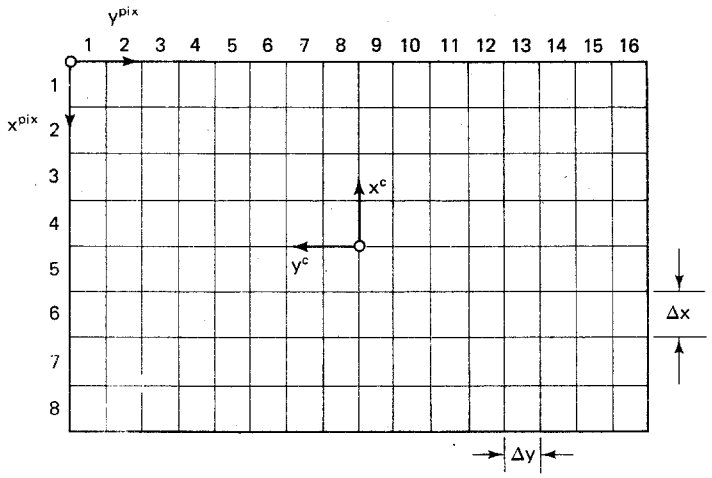


Figure 8-30 An 8 × 16 element sensing array.

assume that the two coordinate frames P and C start out coincident in the center of the sensing array. First we translate P along the x^{pix} axis by $m/2$ and along the y^{pix} axis by $n/2$. This can be represented with the homogeneous translation matrix Tran $[(mi^1 + ni^2)/2]$. Next we rotate P about the z^{pix} axis by π. Since the mobile frame is being rotated about its own unit vector, we must postmultiply by the fundamental homogeneous rotation matrix Rot $(\pi, 3)$. Finally, we must scale the integer pixel values using Scale $(\Delta x, \Delta y, 1, 1)$. Thus the pixel-to-image coordinate transformation matrix is:

$$T_{\text{image}}^{\text{pixel}} = \text{Scale}(\Delta x, \Delta y, 1, 1)\text{Tran}\left(\frac{mi^1 + ni^2}{2}\right)\text{Rot}(\pi, 3)$$

$$= \left[\begin{array}{ccc|c} -\Delta x & 0 & 0 & m\Delta x/2 \\ 0 & -\Delta y & 0 & n\Delta y/2 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{8-7-10}$$

Note that Eq. (8-7-10) can be used to find p^{image} once the row and column of the image of p are known. Given p^{image} and $\lambda = p_3^{\text{camera}}$, we can then apply Prop. 8-7-2 to find the camera coordinates p^{camera}. Finally, knowing p^{camera} and $T_{\text{base}}^{\text{camera}}$, we can determine p^{base}, which is our ultimate objective. In summary:

$$p^{\text{base}} = T_{\text{base}}^{\text{camera}} T_{\text{camera}}^{\text{image}}(\lambda) T_{\text{image}}^{\text{pixel}} p^{\text{pixel}} \tag{8-7-11}$$

Exercise 8-7-2: Pixel-to-Camera Transformation. Use Eq. (8-7-10) and Prop. 8-7-2 to find the 4 × 4 homogeneous transformation matrix $T_{\text{camera}}^{\text{pixel}}(\lambda)$ that maps pixel coordinates into camera coordinates:

$$p^{\text{camera}} = T_{\text{camera}}^{\text{pixel}}(\lambda) p^{\text{pixel}}$$

8-8 STRUCTURED ILLUMINATION

One of the characteristics of robot vision that sets it apart from artificial vision in general is the ability of the user, in most instances, to carefully structure the lighting in the viewing area. Optimal lighting is a very inexpensive way to increase the reliability and the accuracy of a robot vision system. In addition, with the use of structured lighting, three-dimensional information about the object, including a complete height profile, can be obtained.

8-8-1 Light Sources

Perhaps the most effective form of lighting, when it is feasible, is *back lighting*. For a scene that uses back lighting, the illumination comes from behind the scene, so that objects appear as *silhouettes*, as shown in Fig. 8-31. In this case, the objects should be opaque in comparison with the background material. The principal advantage of back lighting is that it produces good contrast between the objects and the background. Consequently, the gray level threshold needed to separate the dark foreground objects from the light background is easily found. Usually, a wide range of gray level thresholds will be effective in segmenting the foreground area from the background.

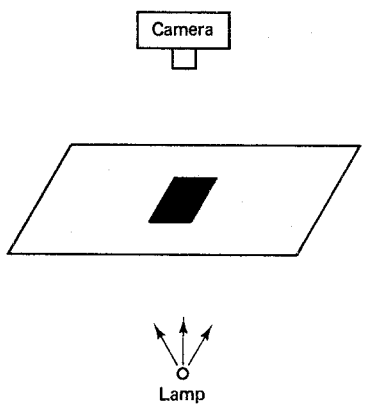


Figure 8-31 Back lighting of a scene.

A light table for back lighting can be constructed by shining one or more lamps onto a diffused translucent surface such as ground glass, diffused plastic, or frosted Mylar. Since only the silhouette of the object is visible to the camera, no direct information about the height of the object is available through back lighting. In fact, back lighting is most effective for inspection of thin, flat objects. More generally, back lighting can be used to inspect objects whose essential characteristics are revealed by profiles generated from one or more physically stable poses. For example, back lighting might be used to distinguish between keys or coins of different sizes, but it cannot be used to distinguish between the heads or tails orientation of an individual coin.

When the outline or silhouette of an object does not provide sufficient information about an object, then some form of *front lighting* must be used. Front lighting is

also necessary when back lighting is simply not feasible, for example, when parts are being transported on an opaque conveyor belt. With front lighting, the light source or sources are on the same side of the scene as the camera, as indicated in Fig. 8-32.

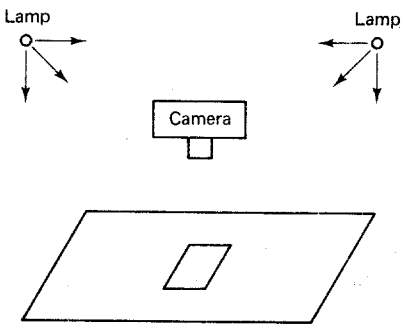


Figure 8-32 Front lighting of a scene.

There are several forms of front lighting which differ from one another in the relative positions and orientations of the camera, the light sources, and the object. Scenes with front lighting are typically low-contrast scenes in comparison with scenes that use back lighting. As a result, considerable care must be taken in arranging the lighting and background to produce a uniformly illuminated scene of maximum contrast. Even then, the gray level threshold needed to separate the foreground area from the background can be quite sensitive. If possible, the background should be selected to contrast sharply with the foreground objects.

The last basic form of lighting is *side lighting*, which can be used to inspect for *surface defects* such as bumps or dimples on an otherwise flat surface. If the lighting is arranged at an acute angle, the defects will be emphasized by either casting shadows or creating reflections, depending upon the surface material. An arrangement of side lighting to highlight surface defects is shown in Fig. 8-33.

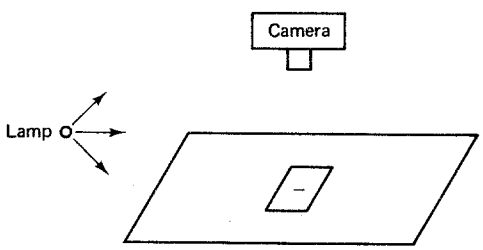


Figure 8-33 Side lighting of a scene.

8-8-2 Light Patterns

The lighting schemes examined thus far are all based on uniform illumination of the entire scene. With the advent of lasers and the use of specialized lenses and optical masks, a variety of *patterns* of light can be projected onto the scene as well. The presence of a three-dimensional object tends to *modulate* these patterns of light when they are viewed from the proper perspective. By examining the modulated light pattern, we can often infer such things as the presence of an object, the object's dimensions, and the orientations of object surfaces.

There is one particularly simple light pattern that can be used to detect the presence of a three-dimensional object and measure its height. Here a *line* or stripe of light is projected onto a scene from the side, say, with a laser and a cylindrical lens, as shown in Fig. 8-34. In this case, the view from the camera will consist of a straight bright line along the x axis if no object is present. When a three-dimensional object is illuminated, a portion of the reflected line will appear below the x axis.

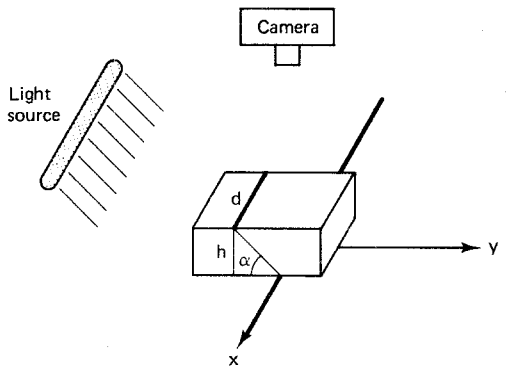


Figure 8-34 Modulation of a line of light by an object.

The distance below the x axis is an indication of the *height* of the object, while the length of the segment appearing below the x axis is an indication of the *width* of the object. If the object is translated along the y axis, or if the object is stationary but the line of light is swept along the y axis, an entire three-dimensional *profile* of the object's height, width, and length can be obtained. To construct a profile, let $y = d(x)$ represent the coordinates of the line of reflected light as seen from directly above by the overhead camera in Fig. 8-34. Note that in general the reflected light pattern $d(x)$ will be a *piecewise-continuous* function of x with $d(x) \leq 0$. Since the angle of the light source is known, we can recover a height profile of the top surface of the object as follows.

Proposition 8-8-1: Height Profile. Let $y = d(x)$ be the observed light pattern in Fig. 8-34, and let α be the angle the light source makes with the horizontal. Then a 3D object is present under the camera if and only if $d(x) < 0$ for some x. In this case, the height of the top surface of the object measured along the line $y = d(x)$ in the xy plane is:

$$h(x) = -(\tan \alpha)\, d(x)$$

This follows directly from Fig. 8-34, using trigonometry. Note that if $d(x)$ is *piecewise-constant*, then the height profile $h(x)$ corresponds to a *vertical cross section* orthogonal to the y axis.

Exercise 8-8-1: Height Measurement. Suppose a line of light is projected onto an object at an angle of $\pi/3$ with respect to the horizontal as shown in Fig. 8-34. The following pattern of reflected light $d(x)$ is obtained from observations with an overhead camera:

$$d(x) = -3x[1(x) - 1(x - 2)]$$

Here $1(x)$ denotes the unit step function. Sketch the height profile of the object along the line $y = d(x)$. Label the height at each end.

8-8-3 Triangulation

A common method of measuring the depth of a particular point on an object is to use range triangles. Consider, for example, the arrangement of a light source and camera shown in Fig. 8-35. Here the light source might be a laser or some other source capable of projecting a narrow beam of light. Let d denote the horizontal distance between the incident light beam and the lens of the camera, and let α be the angle the incident light beam makes with the horizontal. The parameters d and α are fixed and can be measured when the ranging system is installed.

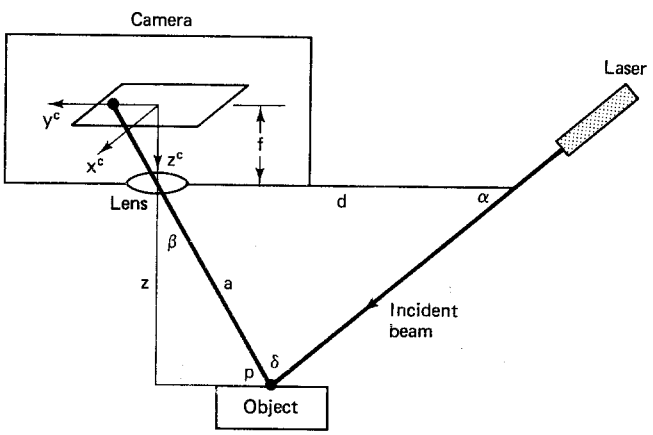


Figure 8-35 Ranging by triangulation.

The incident light beam strikes the object at point p. The coordinates of the image of p with respect to the camera frame $C = \{x^c, y^c, z^c\}$ are denoted p^{image}. Given p^{image} and the effective focal distance of the camera, f, the angle β of the line of sight from the lens to p can be computed using trigonometry. Note that $\pi/2 - \beta \neq \alpha$ in general, because β represents the angle of the line of sight between the camera and the point where the incident beam strikes the object. Therefore β does not represent the angle of reflection. Once β is determined from p^{image} and f, the angle between the incident beam and the line of sight, δ, can be computed, since α is known. Given α, δ, and d, the length of side a can then be determined using the law of sines. Finally, once a has been determined, the vertical distance z can be computed. We summarize the triangulation measurement technique as follows:

Proposition 8-8-2: Triangulation. Let d be the horizontal distance between the camera lens and the incident beam of light, let α be the angle the incident beam makes with the horizontal, and let p^{image} be the coordinates of the image of the point

where the incident beam strikes an object as shown in Fig. 8-35. If f is the effective focal distance of the camera, then the vertical distance z between the lens and the point p can be computed as follows:

$$\beta = \text{atan2}\left(\| p^{\text{image}} \|, f\right)$$

$$z = \frac{\sin \alpha \, (\cos \beta) \, d}{\cos (\alpha - \beta)}$$

Proof. First note from the triangle inside the camera that $\tan \beta = \| p^{\text{camera}} \| / f$, and therefore:

$$\beta = \text{atan2}\left(\| p^{\text{camera}} \|, f\right)$$

Here $\| p^{\text{camera}} \|$ is used, rather than $| p_2^{\text{camera}} |$, because the incident beam does not necessarily lie in the $y^c z^c$ plane. Next, let δ be the angle between the incident beam and the line of sight between the camera and point p. From Fig. 8-35, we see that $\delta = \pi - \alpha - (\pi/2 - \beta) = \beta - \alpha + \pi/2$. Applying the law of sines to the triangle in Fig. 8-35 then yields $(\sin \delta)/d = (\sin \alpha)/a$, and solving for the length of side a, we have:

$$\begin{aligned} a &= d\left(\frac{\sin \alpha}{\sin \delta}\right) \\ &= d\left[\frac{\sin \alpha}{\sin (\beta - \alpha + \pi/2)}\right] \\ &= d\left[\frac{\sin \alpha}{\cos (\beta - \alpha)}\right] \end{aligned}$$

Finally, from inspection of Fig. 8-35, we see that the depth z can be obtained from a and β follows:

$$\begin{aligned} z &= a(\cos \beta) \\ &= \frac{\sin \alpha \, (\cos \beta) \, d}{\cos (\alpha - \beta)} \end{aligned}$$

Prop. 8-8-2 fills in an important missing link in the previous discussion of the inverse perspective transformation. Recall that Prop. 8-7-2 allowed us to compute the coordinates of a point p from the coordinates of the image of p. However, this inverse perspective transformation was based on the assumption that we had a priori knowledge of the depth parameter, $\lambda = p_3^{\text{camera}}$, using a separate measurement technique. One such measurement technique is the triangulation method of Prop. 8-8-2. In particular, we see from Fig. 8-35 that:

$$\lambda = z + f \tag{8-8-1}$$

Exercise 8-8-2: Triangulation. Suppose a light source projecting a narrow beam of light is mounted 60 cm across from the camera at an angle of $\pi/3$ radians with respect to the horizontal as shown in Fig. 8-35. The effective focal distance of the camera is $f = 1$ cm, and the coordinates of the image are $p^{\text{image}} =$

$[0.2, 0.2, 0]^T$. What is the vertical distance between the lens and the point p on the object? Find p^{camera}

8-9 CAMERA CALIBRATION

In robotic applications, the objective is to determine the position and orientation of each part relative to the base frame of the robot. Once this information is known, the proper tool configuration $T_{\text{base}}^{\text{tool}}$ can be selected, and then a joint-space trajectory $q(t)$ can be computed, so as to manipulate the part. With the aid of a robot vision system, we can determine the position and orientation of a part relative to the camera. Thus, to determine the part coordinates relative to the robot base, we must have an accurate transformation from camera coordinates to base coordinates. Experimentally determining this transformation, $T_{\text{base}}^{\text{camera}}$, is called the *camera calibration* problem.

In general, camera calibration requires determining both the position and the orientation of the camera. To illustrate the principle of camera calibration, we restrict our attention to the simpler problem of determining the camera *position*, assuming that the camera orientation is known. Note that it is often not practical to measure the camera position directly, since the camera may not be easily accessible. Furthermore, the required measurements are defined with respect to the origin of the image plane, a point *inside* the camera. Rather than attempt to measure the camera position directly, we instead use the camera itself to generate data from which its position can be inferred. To illustrate this approach, suppose the orientation of the camera frame relative to the robot base frame is fixed as shown in Fig. 8-36. Here the effective focal distance of the camera is f.

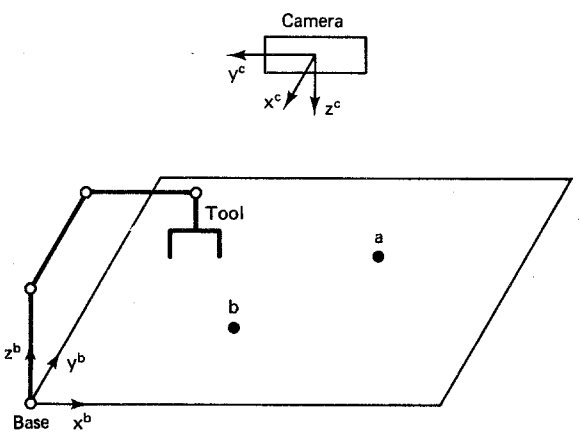


Figure 8-36 Camera calibration.

Let (x_0, y_0, z_0) denote the *unknown* position of the camera frame $C = \{x^c, y^c, z^c\}$ with respect to the base frame $B = \{x^b, y^b, z^b\}$. We see from Fig. 8-36 that the camera-to-base coordinate transformation matrix has the following general form in this case:

$$T_{\text{base}}^{\text{camera}} = \left[\begin{array}{ccc:c} 0 & -1 & 0 & x_0 \\ -1 & 0 & 0 & y_0 \\ 0 & 0 & -1 & z_0 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{8-9-1}$$

To calibrate the camera position, we place a *test pattern* of two points on the work surface, as shown in Fig. 8-36. Let a and b denote the coordinates of the test points with respect to the base frame, and let a^{image} and b^{image} denote the coordinates of the images of the points with respect to the camera frame. To derive the relationship between base coordinates and image coordinates, we transform base coordinates into image coordinates. First note that if we invert the matrix in Eq. (8-9-1), this will map base coordinates into camera coordinates. The perspective transformation summarized in Prop. 8-7-1 then maps camera coordinates into image coordinates. Thus the composition of the two transformations maps base coordinates into image coordinates. In particular, using Prop. 2-4-1, we have:

$$\begin{aligned} T_{\text{image}}^{\text{base}} &= T_{\text{image}}^{\text{camera}} T_{\text{camera}}^{\text{base}} \\ &= T_{\text{image}}^{\text{camera}} [T_{\text{base}}^{\text{camera}}]^{-1} \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & x_0 \\ -1 & 0 & 0 & y_0 \\ 0 & 0 & -1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 & y_0 \\ -1 & 0 & 0 & x_0 \\ 0 & 0 & -1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -1 & 0 & y_0 \\ -1 & 0 & 0 & x_0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/f & (f - z_0)/f \end{bmatrix} \end{aligned} \tag{8-9-2}$$

We can now equate the image coordinates a^{image} with the inferred image coordinates, $T_{\text{image}}^{\text{base}} a$. This can also be done for test point b, and the result, after simplification, is:

$$f(y_0 - a_2) = a_1^{\text{image}}(f - z_0) \tag{8-9-3}$$

$$f(x_0 - a_1) = a_2^{\text{image}}(f - z_0) \tag{8-9-4}$$

$$f(y_0 - b_2) = b_1^{\text{image}}(f - z_0) \tag{8-9-5}$$

$$f(x_0 - b_1) = b_2^{\text{image}}(f - z_0) \tag{8-9-6}$$

These four equations in the unknown camera position coordinates can now be solved simultaneously. Once we have determined (x_0, y_0, z_0), the camera-to-base co-

ordinate transformation matrix in Eq. (8-9-1) is then known. We summarize the camera calibration procedure as follows.

Proposition 8-9-1: Camera Calibration. Consider the robotic workstation shown in Fig. 8-36. Let a and b be the base frame coordinates of two test points on the work surface, and let a^{image} and b^{image} be the coordinates of the images of the test points with respect to the camera frame with $a_1^{\text{image}} \neq b_1^{\text{image}}$. If f is the effective focal distance of the camera, then the camera-to-base coordinate transformation matrix $T_{\text{base}}^{\text{camera}}$ is as given in Eq. (8-9-1), where the position of the camera is:

$$z_0 = f\left[1 + \frac{(a_2 - b_2)}{a_1^{\text{image}} - b_1^{\text{image}}}\right]$$

$$y_0 = a_2 + \frac{(f - z_0)a_1^{\text{image}}}{f}$$

$$x_0 = a_1 + \frac{(f - z_0)a_2^{\text{image}}}{f}$$

Proof. We must solve Eqs. (8-9-3) to (8-9-6) simultaneously for (x_0, y_0, z_0). If we subtract Eq. (8-9-5) from Eq. (8-9-3), this eliminates y_0, and the result is:

$$f(b_2 - a_2) = \frac{a_1^{\text{image}} - b_1^{\text{image}}}{f - z_0}$$

Since $a_1^{\text{image}} \neq b_1{}^{\text{image}}$, this equation can be solved for z_0. The result, after simplification, is:

$$z_0 = f\left(1 + \frac{a_2 - b_2}{a_1^{\text{image}} - b_1^{\text{image}}}\right)$$

Once z_0 is known, the x and y coordinates of the camera frame can then be computed using Eqs. (8-9-3) and (8-9-4), as follows:

$$y_0 = a_2 + \frac{(f - z_0)a_1^{\text{image}}}{f}$$

$$x_0 = a_1 + \frac{(f - z_0)a_2^{\text{image}}}{f}$$

It is of interest to note that even though we have more constraints in Eqs. (8-3-3) to (8-3-6) than unknowns, the constraints are consistent and the extra constraint allows for two alternative formulations of the camera height z_0. The formulation listed in Prop. 8-9-1 is valid as long as $a_1^{\text{image}} \neq b_1^{\text{image}}$. In the event that the test points are chosen so that $a_1^{\text{image}} = b_1^{\text{image}}$, then we can instead subtract Eq. (8-9-6) from Eq. (8-9-4). This eliminates x_0, and the resulting equation can be solved for z_0 to yield:

$$z_0 = f\left[1 + \frac{a_1 - b_1}{a_2^{\text{image}} - b_2^{\text{image}}}\right] \tag{8-9-7}$$

It is clear that as long as the two test points are distinct ($a \neq b$), at least one of the two formulations for z_0 will be well posed, in the sense that it will not involve division by zero.

Example 8-9-1: Camera Calibration

As an example of an application of Prop. 8-9-1, suppose the effective focal distance of the camera is $f = 1$ cm and the two test points generate the following measurements:

$$a = [17, 10, 0]^T$$

$$b = [20, 12, 0]^T$$

$$a^{\text{image}} = [0, \tfrac{-1}{10}, 0]^T$$

$$b^{\text{image}} = [\tfrac{1}{15}, 0, 0]^T$$

In this case, the position of the camera is:

$$z_0 = \left(1 + \frac{-3}{-\frac{1}{10}}\right) = 31 \text{ cm}$$

$$y_0 = 10 + 0 = 10 \text{ cm}$$

$$x_0 = 17 + (-\tfrac{1}{10})(-30) = 20 \text{ cm}$$

Thus the camera-to-base coordinate transformation matrix in this case is:

$$T_{\text{base}}^{\text{camera}} = \left[\begin{array}{rrr:r} 0 & -1 & 0 & 20 \\ -1 & 0 & 0 & 10 \\ 0 & 0 & -1 & 31 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right]$$

Exercise 8-9-1: Camera Calibration. Generalize Prop. 8-9-1 to include the possibility that the test points may not be on the work surface, that is, $(a_3, b_3) \neq (0, 0)$.

8-10 PROBLEMS

8-1. Consider the image $I(k, j)$ and template $T(k, j)$ shown in Fig. 8-37. Using the performance index in Eq. (8-2-2), compute $\rho(x, y)$ for $0 \leq x \leq 2$ and $0 \leq y \leq 1$. What translation of the template produces the best match with the image?

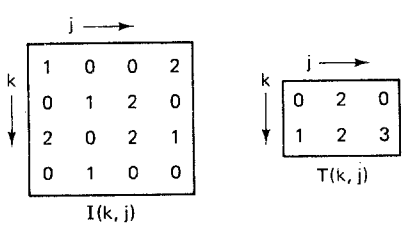


Figure 8-37 An image and a template.

8-2. Repeat Prob. 8-1, but use the normalized cross-correlation performance index $\sigma(x, y)$ in Eq. (8-2-4).

8-3. Consider pixel (k, j) and its neighbors as shown in Fig. 8-38. Derive a numerical approximation $\nabla I(k, j)$ for the gradient of the reflected light intensity, $\nabla i(x, y)$, centered on pixel (k, j). Use averages of first differences of pixels that are adjacent to pixel (k, j). The spacing between adjacent pixels is Δx and Δy. Express your final answer in terms of 3×3 weighting templates.

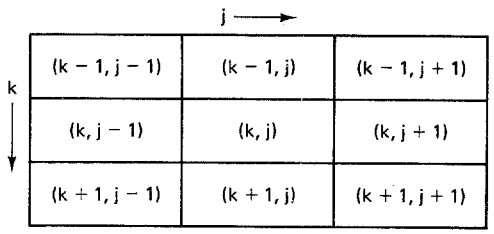


Figure 8-38 Pixel (k, j) and its eight neighbors.

8-4. Write an algorithm that finds the corner points in an $m \times n$ binary image $I(k, j)$ using the corner-point templates in Fig. 8-8.

8-5. Write an algorithm which will identify the boundary of a region by turning all of the interior pixels into background pixels.

8-6. Use the solution to the previous problem to write an algorithm for computing the perimeter of a region in pixels.

8-7. Find expressions for the second-order central moments $\{\mu_{02}, \mu_{11}, \mu_{20}\}$ in terms of the standard moments $\{m_{kj}\}$.

8-8. Consider the binary image $I(k, j)$ in Fig. 8-40, which has a single region R. Compute the following moments of the region R: m_{00}, m_{01}, m_{10}, μ_{02}, μ_{11}, μ_{20}.

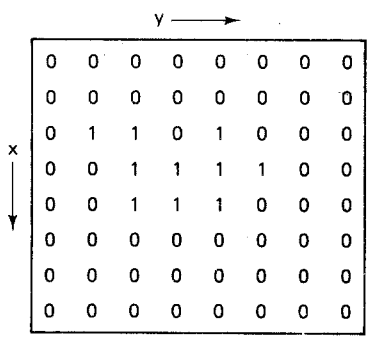


Figure 8-40 A binary image with a single region R.

8-9. Find the area, centroid, and principal angle of the region R in Fig. 8-40.

8-10. The principal angle of a part specifies the orientation of the part. However, the principal angle corresponds to the physical angle needed to grasp the part only if the pixels are equally spaced in both directions. Recall from Eq. (8-3-4) that the ratio of the pixel spacing in the x direction to the pixel spacing in the y direction is called the *aspect ratio* ρ_{xy}. Derive a more general expression for the principal angle which includes the aspect ratio ρ_{xy}. Does your answer reduce to the expression in Def. 8-4-4 when $\rho_{xy} = 1$?

8-11. Let $i(x, y)$ denote an analog image, and let R be a connected region in $i(x, y)$ with an area of A and a perimeter of ρ. Consider the following measure of the *elongation* of R:

$$E(R) \triangleq \frac{A}{\rho^2}$$

(a) Show that $E(R)$ is invariant to translations, rotations, and scaling of R.
(b) Find upper and lower bounds on the value of $E(R)$ that are as tight as possible, and specify regions which correspond to these upper and lower bounds.
(c) Show that $E(R)$ is not a unique measure of shape by sketching two regions R_1 and R_2 of the same size but different shapes for which $E(R_1) = E(R_2)$.

8-12. Find the run-length code Γ of the image in Fig. 8-40 using the scanning sequence: $I(1, 1), I(1, 2), \ldots, I(m, n)$. If each number takes 1 byte to store, what is the percentage saving in space that we gain by encoding this particular image?

8-13. Let a and b represent a pattern of test points as shown in Fig. 8-39. Point a is directly below the camera, and so $a^{\text{image}} = 0$. Point b is at the same elevation as point a, and the distance between point b and point a is d. Given d, b^{image}, and the vertical distance z between point a and the lens, find the effective focal distance f of the camera.

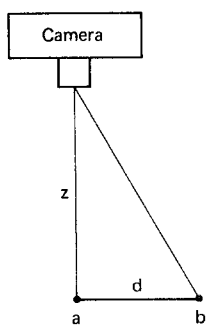


Figure 8-39 Estimating the effective focal length of the camera.

8-14. The position and the orientation of the camera relative to the robot base are specified by the following matrix $T_{\text{base}}^{\text{camera}}$:

$$T_{\text{base}}^{\text{camera}} = \begin{bmatrix} 0 & -1 & 0 & 20 \\ -1 & 0 & 0 & 15 \\ 0 & 0 & -1 & 30 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If the effective focal distance is $f = 1$ cm, find the coordinates of the image of the origin of the base frame of the robot with respect to the camera frame.

REFERENCES

FREEMAN, H. (1961). "On the encoding of arbitrary geometric configurations," *IRE Trans. Electronic Computers,* June, pp. 260–268.

HORN, B. K. P. (1986). *Robot Vision,* MIT Press: Cambridge, Mass.

HU, M. K. (1962). "Visual pattern recognition by moment invariants," *IRE Trans. Information Theory,* February, pp. 179–187.

NEVATIA, R. (1982). *Machine Perception,* Prentice Hall: Englewood Cliffs, N.J.

ROBERTS, L. G. (1968). "Machine perception of three-dimensional solids," pp. 159–197 in J. T. Tippet et al., eds., *Optical and Electro-Optical Information Processing,* MIT Press: Cambridge, Mass.

SNYDER, W. E. (1985). *Industrial Robots: Computer Interfacing and Control,* Prentice Hall: Englewood Cliffs, N.J.

WIDROW, B. (1973). "The rubber-mask technique, I and II," *Pattern Recognition,* Vol. 5, pp. 175–211.

9

Task Planning

High-level planning of robot motion is required in order to achieve the benefits of truly automated production. Planning at this level is referred to as *task planning*. Task planning is concerned more with the general *goals* of the manipulation task than with the specific means used to achieve these goals. The relationships between a task planner and the other parts of a robot control system are shown in block diagram form in Fig. 9-1. Here the numbers inside the blocks indicate the chapters where these subsystems are discussed.

The primary input to the task planner is a task specification supplied by the user. To plan the motions needed to accomplish a task, the task planner uses an internal world model of its environment plus on-line data from sensors, specifically, the vision system. The raw images $I(k, j)$ are first processed by an image analyzer in order to reduce the data to a form that is more readily usable by the task-planning software. Once a specific movement is planned in the form of a discrete tool-configuration trajectory $\{w^k\}$, this information is sent to the trajectory planner. The trajectory planner uses interpolation techniques and the inverse kinematic equations to convert $\{w^k\}$ to an equivalent continuous-time joint-space reference trajectory $r(t)$. The joint-space trajectory then serves as input to the robot controller, which contains a torque regulator that generates the required torque profile $\tau(t)$ for the joints of the robotic arm. There may also be a force and moment sensor mounted at the wrist and tactile sensors mounted at the tool tip. The feedback data, F^{tool}, from these sensors can be used by the robot controller to implement guarded motion and compliant motion as required.

In this chapter we examine the fundamentals of robot task planning by identifying important subproblems and outlining possible solution techniques. The topics covered include task-level programming, the role of uncertainty, configuration-

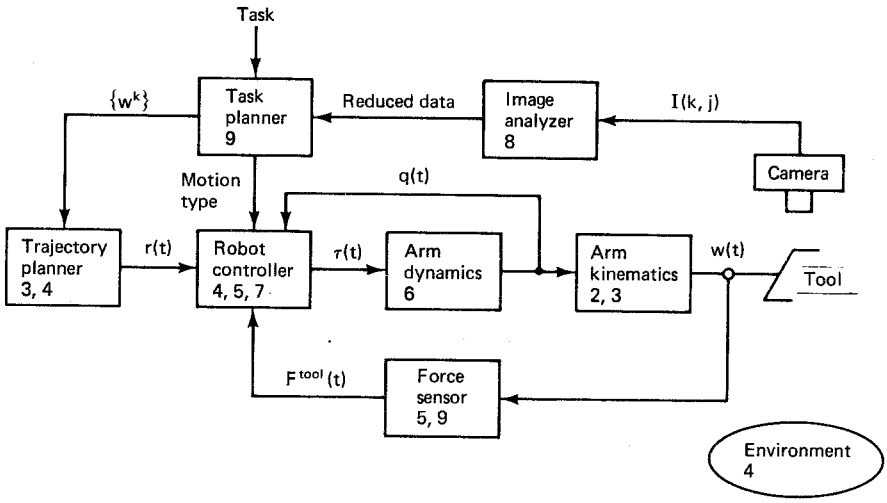


Figure 9-1 Task planner.

space methods, gross-motion planning, grasp planning, fine-motion planning, and simulation techniques. The chapter concludes with a formulation of a task-planning problem and an example of a rudimentary task planner. Many of the topics in this chapter are currently under active research in the fields of artificial intelligence (AI) and robotics.

9-1 TASK-LEVEL PROGRAMMING

Robot programming is typically done at a level that requires detailed knowledge of the robot characteristics, the manipulation task, and the environment in which the task is to be performed. There are three general approaches to robot programming. In industrial applications, the most widely used technique of robot programming is to *teach* the robot using either the teach-pendant method or the lead-through method (Deisenroth, 1985). Here the robot is guided through the desired motions manually. The robot controller records the motions, which are later played back and edited, as needed, until the desired motion is obtained. Although robot teaching has proved to be an effective technique for many applications, this conventional form of robot programming suffers from the following inherent drawbacks which limit its potential:

1. It is a technique that can require the human operator to spend considerable time programming the robot. This not only uses valuable operator time, but also takes the robot out of *useful service*.
2. The parts or objects being manipulated must be presented to the robot at the same positions and orientations each time. This sensitivity to parts positioning necessitates the use of expensive *part fixturing* devices which often have to be customized for the task.
3. The teach method is essentially an *open-loop* type of approach that is not well suited to using external sensors such as vision to adapt to *uncertainties* in the environment.

An alternative to the teach method is to program the robot *externally* with a

special *robot programming language* (Gruver et al., 1983; Lozano-Perez, 1983a). A robot programming language is typically a standard computer programming language that has been extended to include commands to control robot motion and perform sensing operations. Using a robot programming language to control a robot places a considerable burden on the programmer, who must provide specific instructions for each movement of the manipulator. Programming a robot externally through a robot programming language does address several of the limitations associated with the teach method. Much of the programming can be done off-line with a graphic simulator of the robot and its work cell (Schilling and White, 1990). Consequently, although considerable human effort may be required, the robot itself does not have to be taken out of productive service for long. It only has to be made available for the final verification of the programmed manipulation.

Perhaps more important, *off-line programming* can make effective use of information from external sensors as long as the robot programming language has commands or routines which support such external devices. Potential sensors include overhead cameras, tactile arrays, and load cells for force and moment sensing. In addition, robot operations can be synchronized with other items of equipment in the work cell such as conveyors, carousels, part feeders, and additional robots. Finally, safety devices such as light curtains and sentry cameras can be monitored to make sure that foreign objects do not enter the workspace.

The main drawback of off-line programming is that it typically requires detailed specification of the manipulation task and the layout of parts in the workspace. For each motion of the robotic tool, coordinate frames which specify the tool-tip position and tool orientation must be supplied either directly or indirectly. In addition, movement attributes such as the speed, acceleration, or contact force must be specified by the programmer.

A third and more recent approach to robot programming is a higher-level technique called task-level programming (Lozano-Perez and Brooks, 1985). Here, a series of *goals* specifying the desired positions and orientations of the objects being manipulated are supplied by the programmer. The programmer does not need to specify the detailed robot operations needed to achieve these goals. Indeed, the task specification is *robot-independent*. However, a detailed *world model,* or knowledge base, characterizing the robot, its environment, and the manipulation task is needed by the task planner, which takes the task specification as input and produces a detailed set of instructions for the robot controller as output. Task-level programming has great potential, but it is still at a relatively early stage of development in comparison with teach programming and off-line programming. In the remainder of this chapter we examine some basic techniques that are useful for task-level programming and identify some important subproblems that must be solved to develop a functional task planner.

9-2 UNCERTAINTY

Task planning is a challenging problem even when our knowledge of the position and orientation of the parts within the workspace is exact. In reality, the variables which represent the part position and orientation will have a *nominal* value plus an

error term which represents uncertainty:

$$v^{\text{exact}} = v^{\text{nominal}} + \Delta v \tag{9-2-1}$$

$$\|\Delta v\| \leq \Delta v^{\max} \tag{9-2-2}$$

Although the error term Δv is unknown, typically we can place a *bound* on it as in inequality (9-2-2). The error bound $\Delta v^{\max}$ might represent tolerances in the size of a machined part, or it might be associated with the error in a sensor such as an overhead camera used to locate a part. Another possibility arises when the robot is used to grasp a part. Here the uncertainty in our knowledge of the exact position and orientation of the tool produces uncertainty in the final configuration of the manipulated part.

Since exact values for the part variables are unknown, task planning must be done using nominal values and error bounds. For gross-motion planning, we can often neglect the error terms as long as the error bounds are not too large. However, if the required task is an assembly type of operation, then *contact* between parts is required. In these cases, a strategy must be devised which allows for successful completion of the task in spite of the uncertainties in the world model.

The notion of uncertainty permeates all phases of task planning. The errors in our model of the environment tend to propagate and grow when a sequence of manipulations associated with a task is performed (Brooks, 1982). Often, sensors such as vision can be used to reduce uncertainty at certain critical stages of a task. However, the uncertainties can never be completely eliminated by sensing, because the sensors themselves introduce some error. As a simple illustration of how uncertainty can influence task planning, consider the problem of deciding where to place a part fixturing device within the workspace of the robot shown in Fig. 9-2.

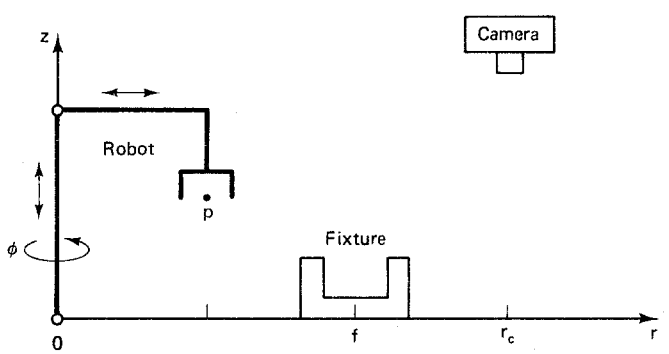


Figure 9-2 Planning the layout of a workstation.

This fixture placement problem is a variation of an example introduced by Brooks (1982). The objective is to insert and remove parts from the work fixture by accurately placing the tool tip p at fixture point f. Let us suppose the manipulator is a cylindrical-coordinate robot with a base-joint precision of $\Delta\phi$, a vertical extension joint precision of Δz, and a radial extension joint precision of Δr. The cylindrical-coordinate robot was analyzed previously in Chap. 1, and from Eq. (1-4-3) the overall precision for positioning the tool tip was found to be:

$$\Delta p^{\text{robot}}(r) = [\Delta r^2 + (r\,\Delta\phi)^2 + \Delta z^2]^{1/2} \tag{9-2-3}$$

Recall that the accuracy with which the tool tip can be placed at an arbitrary location is, at best, half the precision. Thus, to optimize the accuracy with which parts can be inserted and removed from the fixture, it is evident from Eq. (9-2-3) that the distance r between the fixture and the robot should be minimized. In particular, the fixture should be placed at or near the inner surface of the work envelope.

Next, suppose that the location of the part fixturing device is not known directly, but is instead determined using observations from the overhead camera. If we use similar triangles inside and outside the camera, we find that the horizontal displacement of the fixture associated with a 1-pixel change in its image is proportional to the vertical distance between the lens and the fixture. Thus we should raise the fixture as much as possible in addition to placing it near the robot. When we raise the fixture in this manner it will be located along the periphery of the viewing area, where the image can be distorted. This effect is quite pronounced when a very wide angle lens such as a "fisheye" lens is used. To represent the optical distortion, suppose we use the following uncertainty model:

$$\Delta p^{\text{camera}}(r) = \Delta p_0 + \alpha(r - r_c)^2 \tag{9-2-4}$$

Here the first term represents a constant error associated with the fact that the resolution of the image is finite. The second term is a rough model of the distortion in the optics. Here r_c is the radial position of the camera as shown in Fig. 9-2, and α a positive constant.

Notice, that Eq. (9-2-4) specifies that the fixture should be placed directly under the camera to minimize the camera error, while Eq. (9-2-3) specifies that the fixture should be as close to the robot as possible to minimize manipulator error. To find the optimal radial position of the fixture, we must solve Eqs. (9-2-3) and (9-2-4) simultaneously. The solution is displayed graphically in Fig. 9-3.

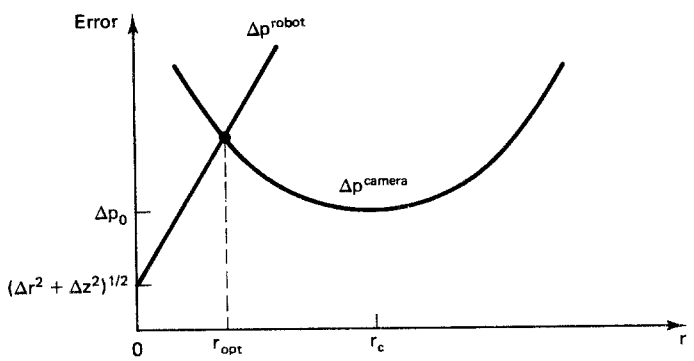


Figure 9-3 Determining the optimal fixture position.

It is clear from Fig. 9-3 that the optimal radial position for the part fixturing device, r_{opt}, lies somewhere between the camera ($r = r_c$) and the robot base ($r = 0$). In the event that the radial precision Δr and the vertical precision Δz of the robot are sufficiently small to be neglected, Eq. (9-2-3) can be approximated as $\Delta p^{\text{robot}} \approx r\,\Delta\phi$. Under these conditions the approximate value for the optimal radial

position of the fixture can be computed algebraically from Eq. (9-2-4) by solving a quadratic equation, the result being:

$$r_{\text{opt}} \approx \frac{2\alpha r_c + \Delta\phi - [(2\alpha r_c + \Delta\phi)^2 - 4\alpha(\alpha r_c^2 + \Delta p_0)]^{1/2}}{2\alpha} \qquad (9\text{-}2\text{-}5)$$

Here the root smaller than r_c is used because this is closest to the robot and therefore generates a smaller common position error, $\Delta p^{\text{robot}}(r_{\text{opt}}) \approx \Delta p^{\text{camera}}(r_{\text{opt}})$.

9-3 CONFIGURATION SPACE

One of the problems that a task planner must address is the gross-motion path-planning problem. Here the objective is to plan a path to move a part from a given *source* position and orientation s to a desired *goal* position and orientation g in the presence of other parts which are regarded as obstacles. The motion is *gross* in the sense that the mobile part is assumed to be free of contact with other parts at both ends of the path. In addition to avoiding collisions with obstacles, the planned path should be optimal or at least near optimal in some sense such as the shortest path or perhaps the path that stays as far away from potential collisions as possible.

The gross-motion path-planning problem has received considerable attention in the literature. In its general form, the path-planning problem requires a search in six-dimensional space, because a moving part has three translational plus three rotational degrees of freedom. Important special cases are investigated by restricting one or more degrees of freedom. For example, arbitrary motion of a part in a plane is a three-dimensional search problem that involves two translational motions within the plane plus one rotational motion about an axis orthogonal to the plane.

A fundamental analytical tool for solving motion-planning problems in general is the configuration-space framework developed by Lozano-Perez (1983b). A *configuration* of a part is a set of parameters which uniquely specify the position of every point on the part, and *configuration space* is the set of all possible configurations. In configuration space the problem of planning the motion of a part through a space of obstacles is transformed into an equivalent, but simpler, problem of planning the motion of a point through a space of enlarged configuration-space obstacles.

9-3-1 Translations

To simplify our discussion of configuration space, we restrict our attention to the motion of polygonal parts within a plane. This means that configuration space will be two-dimensional or three-dimensional depending upon whether or not the mobile polygon is allowed to rotate. As an illustration, consider a scene with two polygonal parts shown in Fig. 9-4.

Suppose triangle A is the mobile part and rectangle B is an obstacle. We assume, initially, that A can translate but not rotate. In general, the mobile part might represent a payload being manipulated by the robot or perhaps the robot itself (Lozano-Perez, 1987). Given a mobile part A, we choose a *reference point* r some-

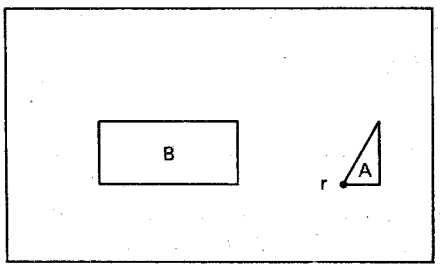


Figure 9-4 A scene with two polygonal parts.

where on the part. To convert to configuration space, we then shrink the mobile part to its reference point r while simultaneously *growing* the obstacles to compensate. In the case of pure translations of the mobile part A, this growth is achieved by sliding part A along the boundary of obstacle B and tracing out the locus of points traversed by the reference point r as illustrated in Fig. 9-5.

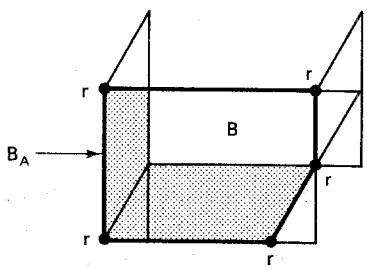


Figure 9-5 Generating the configuration-space obstacle B_A.

As triangle A is shrunk to its reference point r, rectangle B grows into a five-sided polygon, B_A. We refer to the enlarged obstacle B_A as a configuration-space obstacle induced by A. When the original polygonal scene in Fig. 9-4 is redrawn in the configuration space generated by triangle A, the result is as shown in Fig. 9-6. The robot workspace can be regarded as a hole in a large obstacle. Hence the walls of the workspace have grown *inward* by an amount which corresponds to sliding triangle A along the workspace boundary.

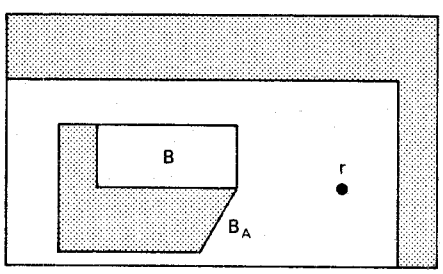


Figure 9-6 Configuration space induced by part A.

As long as the reference point of triangle A stays outside the configuration-space obstacle B_A, triangle A will not collide with obstacle B. Thus we have converted the original problem in Fig. 9-4 of finding a path for a triangle into the equivalent problem in Fig. 9-6 of finding a path for the point r.

Exercise 9-3-1: Configuration Space. Suppose the rectangle B in Fig. 9-4

is the mobile part and the triangle A is an obstacle. Sketch the configuration-space obstacle A_B and reduced workspace assuming the reference point r is the lower right vertex of B.

Note that the configuration-space obstacle B_A in Fig. 9-6 contains the original obstacle B as a subset. Furthermore, the mobile part A and the original obstacle B are both convex, and so is the configuration-space obstacle B_A. These observations hold in general, and furthermore we can place simple bounds on the number of vertices of the configuration-space obstacle, as can be seen from the following result (Lozano-Perez and Wesley, 1979):

Proposition 9-3-1: Configuration Space. Let A and B be convex polygons, and let B_A be the configuration-space obstacle generated by A using reference point r. If n_A, n_B, and n_{BA} denote the number of vertices of A, B, and B_A, respectively, then:

1. $n_B \leq n_{BA} \leq n_A + n_B$.
2. B_A is convex.
3. If $r \in A$, then $B \subseteq B_A$.

Thus the number of vertices in the configuration-space obstacle is bounded from below by the number of vertices in the original obstacle, and from above by the number of vertices in the original obstacle plus the number of vertices in the mobile part. From the scene shown in Fig. 9-6, we see that $4 \leq n_{BA} \leq 7$. In this particular case, $n_{BA} = 5$, because two of the sides of triangle A are parallel with sides of rectangle B.

The observation in Prop. 9-3-1 that when A and B are convex polygons, so is B_A is an important one, because convex polygons are easier to analyze when it comes to detecting collisions between two polygons or between a point and a polygon. There is no loss of generality in assuming that the polygons are convex, because nonconvex polygons can always be synthesized as unions of convex polygons. When the mobile part is modeled as a union of convex polygons, a *common* reference point should be used.

Consider, for example, the scene shown in Fig. 9-7. Suppose that triangle B is an obstacle and that the mobile part A is the union of rectangles A_1 and A_2. We again trace around obstacle B with A_1 and A_2 to generate configuration-space obstacles B_1 and B_2, respectively, as shown in Fig. 9-8. Since the reference point r is a common point for A_1 and A_2, the configuration-space obstacle B_{12} induced by the union of A_1 and A_2 is simply the union of the configuration-space obstacle generated by A_1 and

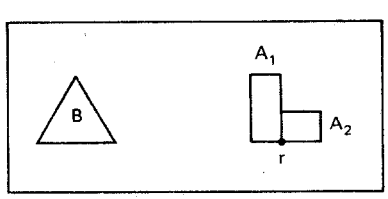


Figure 9-7 A scene with a nonconvex polygon.

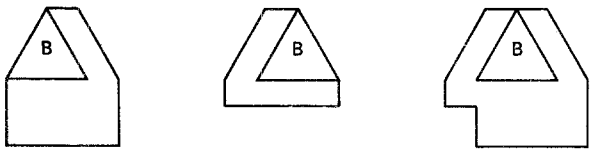


Figure 9-8 Configuration-space obstacles.

the configuration-space obstacle generated by A_2 as shown in Fig. 9-8. That is,

$$B_{12} = B_1 \cup B_2 \tag{9-3-1}$$

The process of computing a configuration-space polygon B_A given polygons A and B can be automated. A program which returns the coordinates of the vertices of B_A given the coordinates of the vertices of A and B is supplied with the Laboratory Manual that accompanies this text (Schilling and White, 1990,). Computing configuration-space obstacles is a key step in solving the gross-motion path-planning problem, as can be seen from the following result (Lozano-Perez and Wesley, 1979):

Proposition 9-3-2: Gross-Motion Path Planning. Let A be a convex mobile polygon with reference point r, and let $\{B^k: 1 \leq k \leq m\}$ be convex polygonal obstacles. Suppose s and g are source and goal points, respectively, for the reference point r. If polygon A does not rotate, then the shortest path from s to g is a piecewise-linear path $\{r^k: 1 \leq k \leq n\}$ whose interior points $(1 < k < n)$ are vertices of configuration-space obstacles $\{B_A^k\ 1 \leq k \leq m\}$.

Thus, as long as the mobile part A does not rotate, the *shortest* path from the source s to the goal g is a piecewise-linear path whose breakpoints are vertices of the configuration-space obstacles induced by A. Since there are a finite number of vertices, they can be efficiently searched for the shortest path using graph theory techniques (Gondran et al., 1984).

As an example of an application of Prop. 9-3-2, consider the scene shown in Fig. 9-9a, where the source and goal values for the reference point are denoted s and g, respectively. The equivalent problem is reformulated in Fig. 9-9b in the configuration space generated by the mobile polygon. Note that two configuration-space obstacles overlap, which means that there is insufficient space for the mobile part to pass between them. The shortest path from s to g in configuration space is shown in Fig. 9-9c. Finally, the motion of the mobile polygon along the shortest path is obtained by transforming from configuration space back to the original space as shown in Fig. 9-9d.

The path in Fig. 9-9 has the mobile polygon sliding along the edges of the obstacles. This is a characteristic feature of the shortest path, a feature which can cause practical problems if there is uncertainty in the position, orientation, size, or shape of the polygons. To avoid the possibility of a collision during the sliding operation, we can place a thin buffer region or envelope around the mobile polygon before generating the configuration-space obstacles, which will now be slightly larger. When this is done, the path will no longer be optimal in the sense of being the shortest path, but it will be near optimal if the buffer region is small.

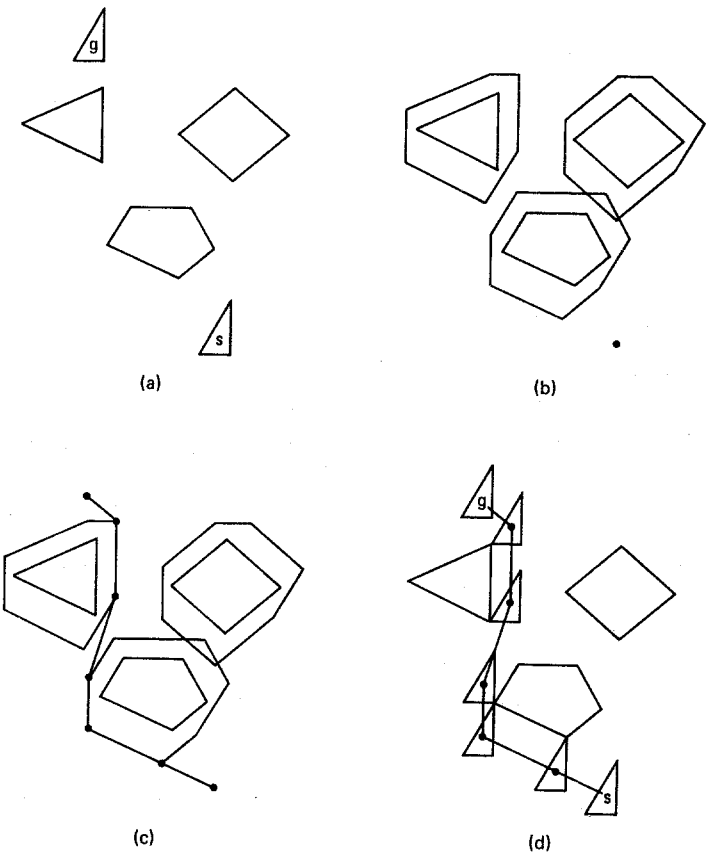


Figure 9-9 Finding the shortest path using configuration space.

9-3-2 Rotations

Configuration space is very effective when applied to purely translational motion in a plane, since the shortest path can readily be found. However, in many instances the mobile object must be rotated if it is to reach the goal. When the mobile polygon is allowed to rotate, the configuration-space obstacles are no longer polygonal. Instead, the surfaces of configuration-space obstacles are curved in the orientation dimension. A simple way to handle rotations is to employ an enlarged polygon which encloses the mobile part over a range of orientations as illustrated in Fig. 9-10.

Suppose the mobile polygon A is allowed to rotate about reference vertex r over a range $[\phi_0, \phi_1]$. If the distance from the reference vertex to the farthest point p on the mobile polygon is d, then the rotated polygon will be contained in a circular sector of radius d and angle $\Delta\phi$, where $\Delta\phi \geq \phi_1 - \phi_0$. The sector can be truncated at the leading and trailing edges of A, and the truncated sector can then be enclosed by a polygon as shown in Fig. 9-10.

The virtue of this approach is that it converts the problem of finding a path for a part that rotates and translates into the simpler problem of finding a path for an en-

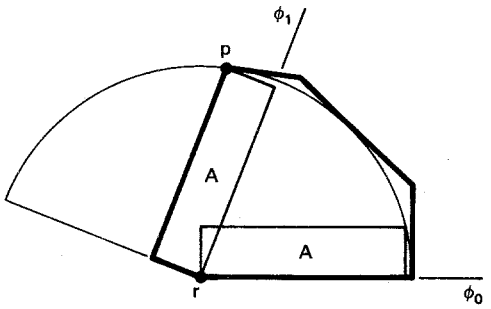
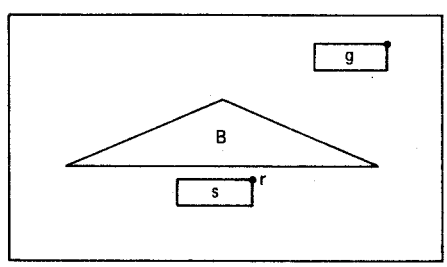


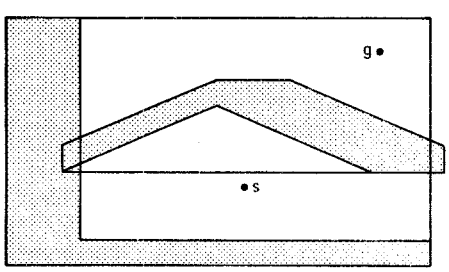
Figure 9-10 Enclosing the rotating part in a convex polygon.

larged part that only translates. If a path is found, then the mobile part can assume *any* orientation in the interval $[\phi_0, \phi_1]$ at each point along the path. The drawback of this simple technique is that it is possible that a path from s to g may not be found when one in fact exists.

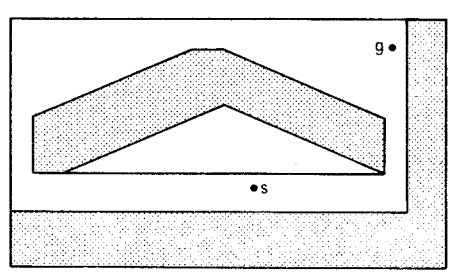
Consider, for example, the problem shown in Fig. 9-11. The original scene in Fig. 9-11a corresponds to a mobile part orientation of $\phi = 0$. Suppose the allowed angles of orientation are $[0, \pi/2]$. The configuration space generated at the initial orientation, $\phi = 0$, is shown in Fig. 9-11b. Note that at this orientation of the mobile rectangle, the configuration-space obstacle is so wide that it overlaps with the workspace boundary, which grows inward. Consequently, there is no path to the goal at this orientation. However, if the mobile rectangle is rotated to $\phi = \pi/2$, then it is possible to move around either side of the obstacle, as can be seen from Fig. 9-11c, which displays the configuration space generated when $\phi = \pi/2$. It is



(a)



(b)



(c)

Figure 9-11 Configuration space for a rotating part.

clear that if we use the simplified technique described in Fig. 9-10 on this problem, no path will be found even though a path does exist.

To address the problems raised by the example in Fig. 9-11, the three-dimensional configuration space $\{x, y, \phi\}$ must be searched directly. One technique is to construct a sequence of configuration-space *slice projections,* that is, configuration-space obstacles corresponding to different values of ϕ (Lozano-Perez, 1983b). Movement within a slice is a pure translation, while movement between slices can be achieved with a pure rotation. The three-dimensional configuration-space obstacles can be approximated using a set of slice projections, and the remaining free space can then be searched for a collision-free path.

Configuration space can be searched by decomposing it into a grid of cells with each cell marked empty, full, or mixed depending upon whether or not a configuration-space obstacle is present (Brooks and Lozano-Perez, 1985). A path search from s to g is performed using only empty cells. If no path is found, a combination of empty and mixed cells is used. The tessellation grid is then made finer by decomposing the mixed cells along the path into smaller cells again empty, full, or mixed. This process is repeated until a path is found through empty cells. This method has been successfully used to solve some very difficult path-planning problems in cluttered workspaces (Brooks and Lozano-Perez, 1985). The main drawback is that the computational effort can sometimes be considerable, particularly for a real-time implementation.

Exercise 9-3-2: Slice Projection. Sketch the configuration-space slice projection of the scene in Fig. 9-4 associated with the orientation angle $\phi = -\pi/2$.

9-4 GROSS-MOTION PLANNING

An alternative to the configuration-space approach to solving the gross-motion path-planning problem is to search the free space directly. Brooks (1983) proposed an explicit representation of free space based on overlapping generalized cones having straight spines and nonincreasing radii. He refers to these representations of the space between obstacles as *freeways*. Translations are performed along freeways and rotations are performed at the intersections of freeways. One useful feature of this method is that it typically generates paths for the mobile part that stay well away from the obstacles, although this means that the paths are somewhat longer than the shortest path. When the workspace is sparsely populated with obstacles, this method is very fast and quite effective. The principal drawback to this approach occurs when the workspace is cluttered with closely spaced obstacles. In this case the method sometimes fails to find a safe path when one exists.

A refinement of the freeway method can be achieved by using an explicit representation of free space based on a generalized Voronoi diagram (Lee and Drysdale, 1981). A generalized Voronoi diagram (GVD) in free space is a locus of points which are equidistant from two or more obstacle boundaries. Once a GVD representation of free space is obtained, the shortest path having an adequate radius is easily found using graph theory techniques (Gondran et al., 1984). Heuristics can then be

employed to perform the actual movement along the chosen GVD path (Takahashi and Schilling, 1989).

9-4-1 Generalized Voronoi Diagrams (GVD)

Generalized Voronoi diagrams can be generated by decomposing the general problem into a number of simpler subproblems. In constructing a GVD, there are three basic types of interaction, as illustrated in Fig. 9-12. The first type of interaction is between a *pair of edges*, as shown in Fig. 9-12a. Here P_3P_4 is an edge interacting with edge P_1P_2. The parameter λ represents the *distance* along edge P_1P_2 measured from P_1. The *radius* of the GVD along P_1P_2 can be expressed by the following piecewise-linear function of λ, where sgn denotes the signum function and d, l_0, l_1, and l_2 are as shown in Fig. 9-12a:

$$R(\lambda) = \frac{(\lambda - l_0)(l_0 - l_1 + \text{sgn}\,(\lambda - l_0)l_2)}{d} \tag{9-4-1}$$

The second type of interaction is between a *vertex and an edge*, as illustrated

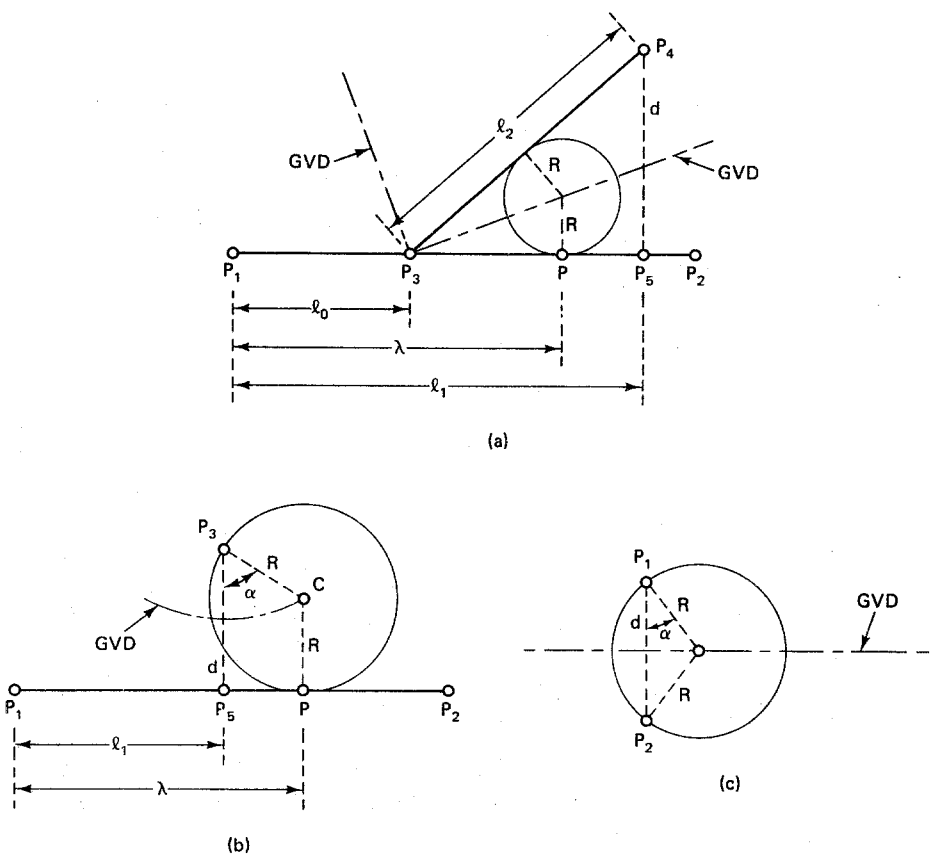


Figure 9-12 Three basic types of interaction in a GVD.

in Fig. 9-12b. Here P_1P_2 is an edge and P_3 is a vertex located a distance d from the edge. Again, if λ represents the distance along edge P_1P_2 measured from P_1, then the radius of the GVD along P_1P_2 in this case can be expressed by the following parabolic function of λ, where l_1 is as shown in Fig. 9-12b:

$$R(\lambda) = \frac{d^2 + (\lambda - l_1)^2}{2d} \tag{9-4-2}$$

A GVD between an edge and a vertex can also be expressed relative to the vertex. In particular, if α represents the *angle* about the vertex measured counterclockwise from a line normal to the edge as shown in Fig. 9-12b, then the radius of the GVD about P_3 can be expressed:

$$R(\alpha) = \frac{d}{1 + \cos \alpha} \tag{9-4-3}$$

The third type of interaction is between a *pair of vertices,* as illustrated in Fig. 9-12c. Here P_1 and P_2 are vertices separated by a distance d. The GVD, in this case, is a straight line midway between the vertices and normal to the line joining them. In terms of the angle α about vertex P_1, the radius of the GVD about P_1 can be expressed:

$$R(\alpha) = \frac{d}{2\cos \alpha} \tag{9-4-4}$$

More complex generalized Voronoi diagrams are constructed using combinations of the three basic GVD types. As an illustration, consider the problem of finding a GVD along a *reference edge* P_1P_2 induced by a second edge P_3P_4 as shown in Fig. 9-13. Here it is assumed that the edges are skew and that P_3 is the end point of edge P_3P_4 that is closest to the line generated by edge P_1P_2.

The reference edge P_1P_2 is decomposed into a number of contiguous *segments,* each corresponding to a different basic GVD type. Over each segment, the GVD is either a *linear* curve between two edges or a *parabolic* curve between an edge and an end-point vertex. The types of interaction that appear in the GVD in Fig. 9-13 are summarized in Table 9-1.

For each segment of the GVD in Fig. 9-13, the radius R can be expressed as a function of λ, the distance from P_1 along edge P_1P_2. Note that the expressions for the radius in Eqs. (9-4-1) and (9-4-2) are both special cases of the following *generic* radius equation, where L_0 represents the length of the reference edge P_1P_2:

$$R(\lambda) = a(\lambda)\lambda^2 + b(\lambda)\lambda + c(\lambda) \qquad 0 \leq \lambda \leq L_0 \tag{9-4-5}$$

It follows that the radius of a GVD along an edge can be represented by a *piecewise-quadratic* function. The parameters $\{a(\lambda), b(\lambda), c(\lambda)\}$ in Eq. (9-4-5) are piecewise-constant functions whose values change at the segment boundaries. The segment boundaries and the parameter values over each segment of the GVD in Fig. 9-13 are summarized in Table 9-2.

The parameters appearing in Table 9-2 are all defined with respect to Fig. 9-13. If the point E_3 represents the intersection of the lines generated by edges P_1P_2 and P_3P_4, then L_0 is the length of edge P_1P_2, L_1 is the length of edge P_3P_4, and L_2 is the distance between points E_3 and P_3. Next let F_1 be the orthogonal projection of

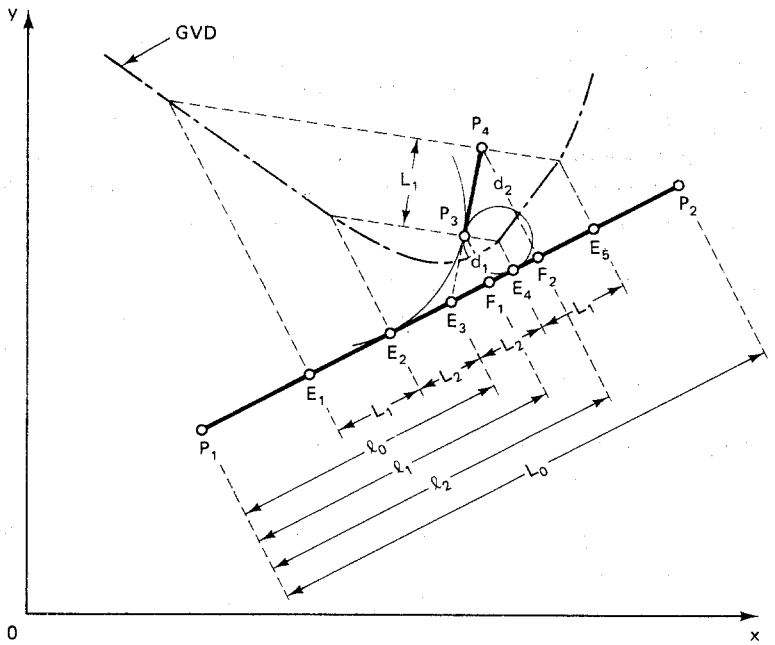


Figure 9-13 GVD induced by a skew edge.

TABLE 9-1 GVD ALONG EDGE P_1P_2 INDUCED BY EDGE P_3P_4

Segment	Interaction	Curve
P_1E_1	Edge P_1E_1 and vertex P_4	Parabolic
E_1E_2	Edge E_1E_2 and edge P_3P_4	Linear
E_2E_4	Edge E_2E_4 and vertex P_3	Parabolic
E_4E_5	Edge E_4E_5 and edge P_3P_4	Linear
E_5P_2	Edge E_5P_2 and vertex P_4	Parabolic

TABLE 9-2 PARAMETERS OF A GVD BETWEEN TWO SKEW EDGES

Segment	λ	$a(\lambda)$	$b(\lambda)$	$c(\lambda)$
P_1E_1	$[0, l_0 - L_1 - L_2]$	$0.5/d_2$	$-l_2/d_2$	$0.5(d_2^2 + l_2^2)/d_2$
E_1E_2	$[l_0 - L_1 - L_2, l_0 - L_2]$	0	$(l_0 - L_1 - L_2 - l_2)/d_2$	$-l_0b(\lambda)$
E_2E_4	$[l_0 - L_2, l_0 + L_2]$	$0.5/d_1$	$-l_1/d_1$	$0.5(d_1^2 + l_1^2)/d_1$
E_4E_5	$[l_0 + L_2, l_0 + L_1 + L_2]$	0	$(l_0 + L_1 + L_2 - l_2)/d_2$	$-l_0b(\lambda)$
E_5P_2	$[l_0 + L_1 + L_2, L_0]$	$0.5/d_2$	$-l_2/d_2$	$0.5(d_2^2 + l_2^2)/d_2$

the end point P_3 onto the line generated by edge P_1P_2, and let d_1 be the distance between points P_3 and F_1. Similarly, let F_2 be the orthogonal projection of the end point P_4 onto the line generated by edge P_1P_2, and let d_2 be the distance between points P_4 and F_2. Then l_0 is the distance from P_1 to E_3, l_1 is the distance from P_1 to F_1, and l_2 is the distance from P_1 to F_2. Note that the constants $\{l_0, l_1, l_2\}$ can be negative. The construction in Table 9-2 represents a general case. Depending upon the relative values of the parameters, one or more of the five segments listed in Table 9-2 may lie outside the range $0 \leq \lambda \leq L_0$.

An important special case of Fig. 9-13 arises when vertex P_3 lies on edge P_1P_2. This will occur when two obstacles touch, when an obstacle touches the workspace boundary, or when the pair of edges belong to the same obstacle. In these cases the construction in Table 9-2 simplifies, with $E_3 = P_3$ and $L_2 = 0$. Thus segment E_2E_4 shrinks to a point and Table 9-2 is reduced to four rows. If edge P_3P_4 is also sufficiently long, then points E_1 and E_5 lie outside edge P_1P_2, in which case the representation in Table 9-2 reduces to the basic piecewise-linear GVD type in Eq. (9-4-1).

The formulation in Table 9-2 is based on the assumption that the two edges are skew. For the special case when edge P_3P_4 is parallel to edge P_1P_2, Table 9-2 simplifies. In this case the point E_3 and the distance l_0 are undefined because the lines generated by edges P_1P_2 and P_3P_4 do not intersect. The segment boundaries and the parameter values over each segment of the GVD when the edges are parallel are summarized in Table 9-3.

TABLE 9-3 PARAMETERS OF A GVD BETWEEN TWO PARALLEL EDGES

Segment	λ	$a(\lambda)$	$b(\lambda)$	$c(\lambda)$
P_1F_1	$[0, l_1]$	$0.5/d_2$	$-l_1/d_1$	$0.5(d_1^2 + l_1^2)/d_1$
F_1F_2	$[l_1, l_2]$	0	0	$0.5d_1$
F_2P_2	$[l_2, L_0]$	$0.5/d_2$	$-l_2/d_2$	$0.5(d_2^2 + l_2^2)/d_2$

Note that $d_2 = d_1$, because the two edges are parallel. As the length of edge P_3P_4 shrinks to zero, edge P_3P_4 becomes vertex P_3, in which case $l_2 = l_1$. Fig. 9-13 then reduces to Fig. 9-12b, and the representation in Table 9-3 reduces to the basic parabolic GVD type in Eq. (9-4-2).

Given a method for constructing the GVD along a reference edge P_1P_2 induced by another edge P_3P_4, it remains to construct the *composite* GVD along P_1P_2 induced by the edges of all of the obstacles in the workspace. This is achieved by sequentially comparing pairs of GVD lines and synthesizing from them a new GVD line containing the segments closest to the reference edge, P_1P_2.

Each GVD along an edge of an obstacle is expressed in terms of a radius function. This radius information is useful when the GVD is searched for a safe path for the mobile part. To represent the entire GVD in terms of a single global coordinate frame, the coordinates of the points on the GVD must be determined. Suppose (x_1, y_1) and (x_2, y_2) denote the coordinates of end points P_1 and P_2, respectively. The coordinates of the locus of points which characterize the GVD along edge P_1P_2 can then be expressed in terms of the radius function as follows:

$$x(\lambda) = x_1 + \frac{(x_2 - x_1)\lambda}{L_0} - \frac{(y_2 - y_1)R(\lambda)}{L_0} \qquad 0 \le \lambda \le L_0 \qquad (9\text{-}4\text{-}6a)$$

$$y(\lambda) = y_1 + \frac{(y_2 - y_1)\lambda}{L_0} + \frac{(x_2 - x_1)R(\lambda)}{L_0} \qquad 0 \le \lambda \le L_0 \qquad (9\text{-}4\text{-}6b)$$

The formulation in Eq. (9-4-6) follows from basic trigonometry. Note that the signs of the last terms are based on the convention that the edges of the obstacles are traversed in a clockwise sense.

In order to complete the GVD representation of free space, GVD lines must also be constructed about obstacle vertices. Each vertex P_k has an adjacent pair of edges $P_{k-1}P_k$ and P_kP_{k+1} associated with it. Since GVD lines have already been constructed for these edges, the range of angles about vertex P_k over which a GVD must be computed is easily determined. In particular, let α_0 be the angle that the normal to edge P_kP_{k+1} makes with the x axis, and let α_1 be the angle that the normal to edge $P_{k-1}P_k$ makes with the x axis. Then the *range* of angles over which the GVD about vertex P_k must be computed is:

$$A = [\alpha_0, \alpha_1] \qquad (9\text{-}4\text{-}7)$$

Note that in some instances it is possible that $\alpha_1 \le \alpha_0$, in which case sector A is the empty set. This occurs when a vertex is *concave*. Here the GVD lines along edges adjacent to the vertex overlap. Consequently, there is no need to construct GVD lines about concave vertices. The size of the sector over which a GVD must be constructed is max $\{0, \pi - \beta\}$, where β is the interior angle of the vertex.

Recall that there are two basic types of generalized Voronoi diagrams about a vertex: a GVD between a vertex and an edge, as shown in Fig. 9-12b; and a GVD between two vertices, as shown in Fig. 9-12c. In the general case, a GVD about a vertex is decomposed into a number of contiguous *sectors*, each sector corresponding to a different basic GVD type. The expressions for the radius of a GVD about a vertex in Eqs. (9-4-3) and (9-4-4) are both special cases of the following *generic* radius equation:

$$R(\alpha) = \frac{b(\alpha)}{\cos[\alpha - a(\alpha)] + c(\alpha)} \qquad \alpha_0 \le \alpha \le \alpha_1 \qquad (9\text{-}4\text{-}8)$$

It follows that the radius of a GVD about a vertex can be represented by Eq. (9-4-8) where $\{a(\alpha), b(\alpha), c(\alpha)\}$ are piecewise-constant functions that change values at the sector boundaries. Once the sector boundaries have been determined, it is a simple matter to evaluate the GVD parameters. For the case of a GVD between a vertex and an edge illustrated in Fig. 9-12b, a is the angle between the x axis and the line P_3P_5, b is the length of line P_3P_5, and $c = 1$. For the case of a GVD between two vertices illustrated in Fig. 9-12c, a is the angle between the x axis and the line P_1P_2, b is half the length of P_1P_2, and $c = 0$. Thus the key step is the determination of the sector boundaries.

To illustrate the computation of sector boundaries, consider the case shown in Fig. 9-14. Note that by using Eq. (9-4-6), the coordinates of the end points D_1 and D_4 can be found from the expressions for the GVD lines along the edges adjacent to

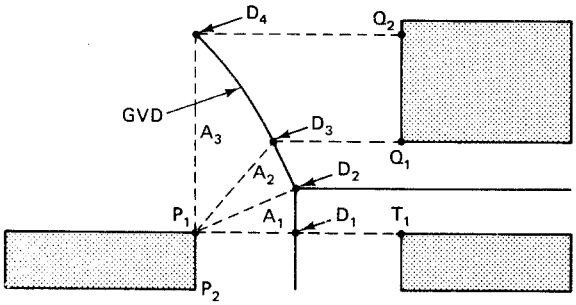


Figure 9-14 Determining sector boundaries.

vertex P_1. This provides the lower and upper limits, α_0 and α_1, respectively. Next the interaction between vertex P_1 and each of the edges of the workspace obstacles is examined. At this point, there is no need to directly construct the GVD lines about P_1 induced by each of the edges and then synthesize from them a composite GVD in the range $[\alpha_0, \alpha_1]$. Instead, one can simply search through the previously constructed data structures associated with the GVD lines along obstacle edges to find those segments that were induced by interaction with vertex P_1. In the case illustrated in Fig. 9-14, sector A_3 would be identified in this manner because it has previously been constructed by the interaction between vertex P_1 and edge Q_1Q_2. The coordinates of the point D_3 can then be computed from the GVD along Q_1Q_2. From this information, the sector boundary for A_3 and the GVD parameters $\{a_3, b_3, c_3\}$ are determined. In particular, if T_1 is the orthogonal projection of P_1 onto the line generated by edge Q_1Q_2, then a_3 is the angle that the line P_1T_1 makes with the x axis, b_3 is the length of line P_1T_1, and $c_3 = 1$.

After all of the interactions between vertex P_1 and obstacle edges have been investigated, the GVD about P_1 in the range $[\alpha_0, \alpha_1]$ may still not be complete. In the case of Fig. 9-14, it remains to find the GVD between the points D_1 and D_3. The only type of interaction left to consider is between two vertices. Recall from Fig. 9-12c that a GVD between two vertices is simply a straight line. At this point, generalized Voronoi diagrams are examined between vertex P_1 and the remaining obstacle vertices until a GVD is found which passes through the point D_1. In the case illustrated in Fig. 9-14, vertex T_1 is paired with P_1 and a GVD is constructed between P_1 and T_1. Next a check is made to see whether the GVD between P_1 and T_1 passes through the end point D_3. If it does, the GVD about P_1 between D_1 and D_3 is complete. Otherwise, the procedure searches for an obstacle vertex for which the GVD with P_1 crosses the GVD between P_1 and T_1 within the angle $D_1P_1D_3$. Let the intersection of these two GVD lines be denoted D_2, as in Fig. 9-14. If more than one vertex is found with this property, the procedure chooses the vertex which minimizes the angle $D_1P_1D_2$. In the case of Fig. 9-14, the vertex Q_1 is chosen. The procedure next checks to see whether the new GVD between P_1 and Q_1 passes through the end point D_3. In this case it does, and so the GVD is complete. Otherwise, the last step of the procedure is repeated.

As an example of a GVD constructed by this approach, consider the layout of three obstacles shown in Fig. 9-15. Some of the sections of the GVD are segments along an edge of an obstacle, while others are sectors about a vertex of an obstacle. Note that each section of the GVD can be expressed in two distinct, but equivalent,

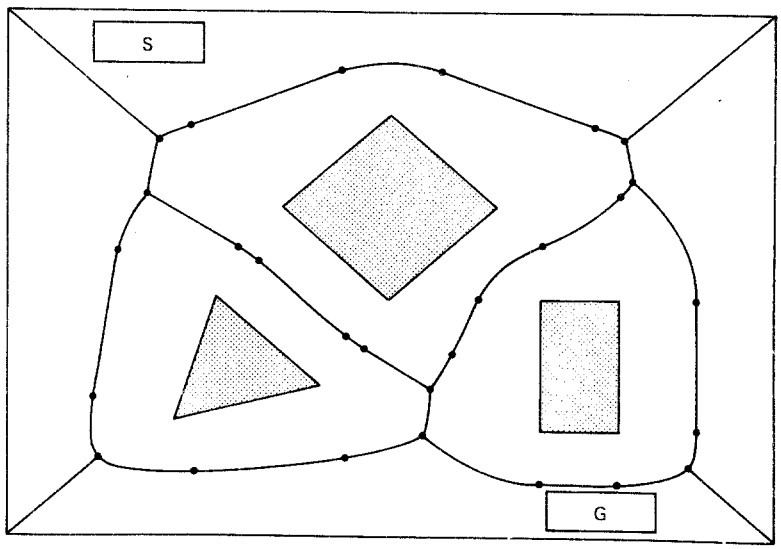


Figure 9-15 A complete GVD.

ways, depending upon which member of the interacting pair is taken as the reference edge or vertex.

9-4-2 Motion Heuristics

A GVD representation of free space is constructed in order to simplify the search for a collision-free path for the mobile part (O'Dunlaing and Yap, 1985). In order to conduct an efficient search, the GVD is first converted to an equivalent *graph* of nodes and arcs. There are three types of nodes that arise in the GVD graph: junction nodes, which are generated when three or more GVD lines intersect: terminal nodes, which correspond to dead ends of GVD lines; and pseudo-nodes. Pseudo-nodes are *source* and *goal nodes* inserted in the graph near the source and goal points of the mobile part. They serve as entry and exit points on the graph.

The *arcs* of the graph are GVD lines connecting pairs of nodes. An arc consists of a sequence of piecewise-linear sections between *via-points,* each via-point being characterized by the parameters (x, y, R), where (x, y) are the coordinates of the via-point and R is the radius of the GVD at the via-point. Smooth parabolic GVD curves can be approximated arbitrarily closely with piecewise-linear arcs by controlling the spacing between the via-points.

Standard graph theory techniques (Gondran et al., 1984) can be employed to search for a path from the source node to the goal node. One starts by examining the nodes adjacent to the source node. Nodes adjacent to these nodes are then investigated, and the search continues to expand all branches as subgraphs until each subgraph reaches a dead end or the goal node. If the minimum radius of an arc is found to be less than half the width of the moving part, a collision is inevitable. Consequently, these narrow arcs are regarded as dead ends. When a node adjacent to subgraph A is already reached by subgraph B and the total length for subgraph A exceeds

the length for subgraph B, subgraph A is discarded. The shortest subgraph which reaches the goal node is taken as the candidate for the shortest collision-free path for the mobile part. An example of a path found by this method is shown in Fig. 9-16.

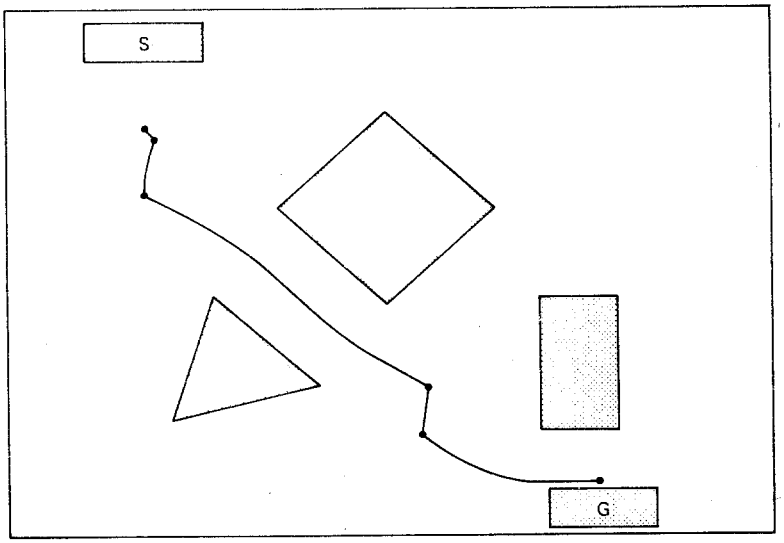


Figure 9-16 The shortest GVD path.

Given a candidate for the shortest collision-free path from the source node to the goal node, it remains to plan the actual motion of the mobile part along the path. Here the details of the size and shape of the mobile part must be exploited in order to develop heuristic techniques for executing the movement. To simplify these motion heuristics, we assume that the mobile part is a *rectangle* or can be enclosed by a rectangular hull. This is not a severe restriction for convex polygonal parts. Indeed, if the part is approximately circular in shape, then the path-planning problem simplifies, because the orientation of the part is no longer an important factor. Thus the more interesting case occurs when the part is elongated, and it is here that a rectangular hull is often an effective approximation of the actual object size and shape.

A simple motion heuristic can be used for paths that have a radius somewhat larger than half the width of the mobile part. Let W and L denote the *width* and *length,* respectively, of the mobile rectangle, where $W \leq L$. Two reference points, P_1 and P_2, are placed on the part along its major axis a distance $L/4$ from each end, as shown in Fig. 9-17.

The basic idea behind this *wide-path* motion heuristic is to have the two reference points of the moving part trace the GVD path much like the front and rear wheels of an automobile. The head point P_2 is advanced incrementally along the GVD, and the corresponding location for the tail point P_1 is determined by finding the intersection of the GVD with a circle of radius $L/2$ centered at P_2. To determine when the wide-path motion heuristic is applicable, consider the two circles about P_1 and P_2 in Fig. 9-17. Note that the entire rectangular object is contained within the two overlapping circles. The wide-path motion heuristic is therefore attempted when

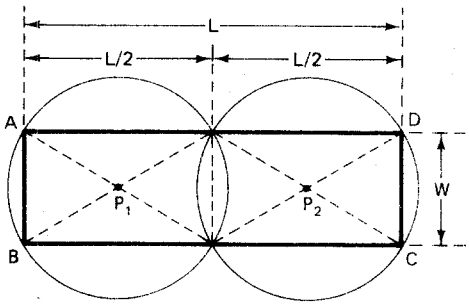


Figure 9-17 The wide-path motion heuristic.

the minimum radius of the chosen path is larger than the radius of the circles:

$$R > \left(\frac{W^2}{4} + \frac{L^2}{16}\right)^{1/2} \tag{9-4-9}$$

When the chosen path satisfies the minimum-radius constraint in inequality (9-4-9), this technique for moving the part along the chosen path works effectively as long as the spacing between via-points is sufficiently small (Takahashi and Schilling, 1989). An example of the motion planned with the wide-path heuristic is shown in Fig. 9-18. Note that the moving part stays well away from the obstacles and exhibits smooth movement, in the sense that it rotates as it translates.

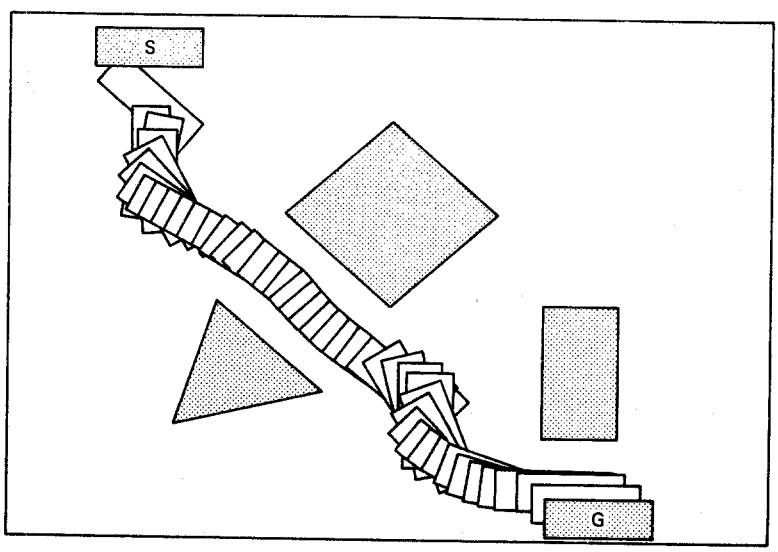


Figure 9-18 Motion planned with the wide-path heuristic.

Once motion along a path is planned, it should be *validated* to see whether there are any collisions. If a collision is detected, a wider path might be attempted or else closer spacing between troublesome via-points might be employed. Alternatively, a more sophisticated motion heuristic can be applied to the original path. This is required, for example, if the only available path satisfies $R > W/2$ but fails to sat-

isfy constraint (9-4-9). In these cases it is useful to develop a local representation of free space so the mobile part can slide to one side of the GVD as needed to round a tight corner or pass through a narrow gap. An example of a path planned in a cluttered workspace using more sophisticated motion heuristics (Takahashi and Schilling, 1989) is shown in Fig. 9-19.

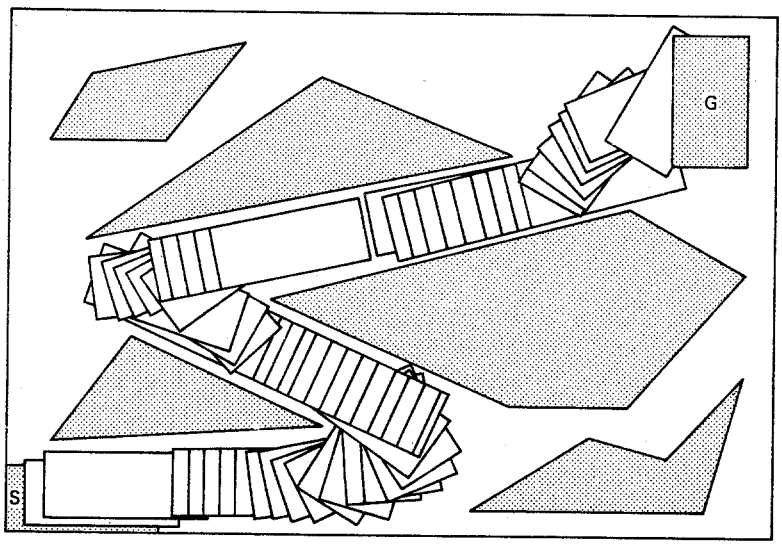


Figure 9-19 A path planned in a cluttered workspace.

Like the freeway method, the GVD technique computes rotations quite efficiently and therefore generates fast run times. Note that the final motion is quite smooth, in the sense that it follows curved GVD lines, stays well away from the obstacles when possible, and rotates as it translates, not just at isolated points in the workspace.

9-5 GRASP PLANNING

The *grasp-planning problem* is one of determining a configuration for the robotic tool that produces a safe, reachable, and secure grasp of the part at both ends of the path along which it is to be transported (Lozano-Perez and Brooks, 1985; Wolter et al., 1985). A *safe* grasp configuration is one which does not cause the robot to come into contact with other parts in the neighborhood of the part being manipulated. Thus, if the part to be manipulated is adjacent to another part, say, in a part feeder, then the adjacent face is clearly not a feasible grasping surface.

A *reachable* grasp configuration is one which is within the work envelope of the manipulator, and one for which a collision-free path to the grasp configuration is available (Peshkin and Sanderson, 1986). In grasping a part at the initial configuration, consideration must be given to how the part will be transported and set down at its final configuration. For example, consider the problem of inverting a block that rests on a horizontal work surface as shown in Fig. 9-20. Normally we would grasp

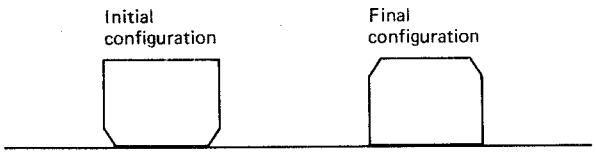


Figure 9-20 Grasping a part.

the part from above with the approach vector r^3 orthogonal to the work surface. This works fine for grasping the part, but then there is no way to set the part down with the block inverted without having the tool collide with the work surface. Consequently in planning the grasp of a part we must *look ahead* to anticipate what will be done with the part after it is grasped. For the problem in Fig. 9-20 it would be better to grasp the part from the side, if possible, and then roll the tool by π before setting the part down. If this is not feasible, then the part might be grasped at an angle of $\pi/4$ and then temporarily set down at an angle of $-\pi/4$. This *intermediate configuration* would go halfway toward inverting the block. The process could then be repeated, thereby transforming the block from the intermediate configuration to the final configuration.

A *secure* or stable grasp is one for which the part will not move or slip during part transfer or subsequent part mating operations. For example, an attempt might be made to place the center of mass of the manipulated part on an axis between the fingertips, thus minimizing the torques on the fingertip surfaces.

Grasp planning is a difficult facet of task planning, because there are many different types of tools and many different shapes and textures of parts to be grasped. Perhaps the most general type of tool is a *parallel-jaw* gripper, where the fingertip surfaces used to contact the part remain parallel as the tool opens and closes. One design for a parallel-jaw gripper is based on a *parallel bar* mechanism, as shown in Fig. 9-21. The Rhino XR-3 robot uses this type of tool.

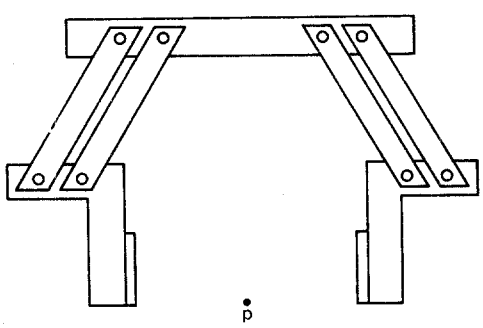


Figure 9-21 A parallel-jaw gripper based on parallel bars (Rhino XR-3).

The parallel bar mechanism has the advantage that the maximum physical width of the tool is relatively narrow, not too much larger than the maximum tool opening. However, as the tool opens and closes, the fingertips extend and retract slightly in order to remain parallel. Thus the tool-tip position p moves in and out along the approach vector when the tool is activated.

An alternative approach to designing a parallel-jaw gripper is to use a *rack-and-pinion* mechanism similar to that found in the steering of many automobiles. A

picture of a commercial parallel-jaw gripper based on a rack-and-pinion design, the Intelledex 660 Small Servo Tool, is shown in Fig. 9-22.

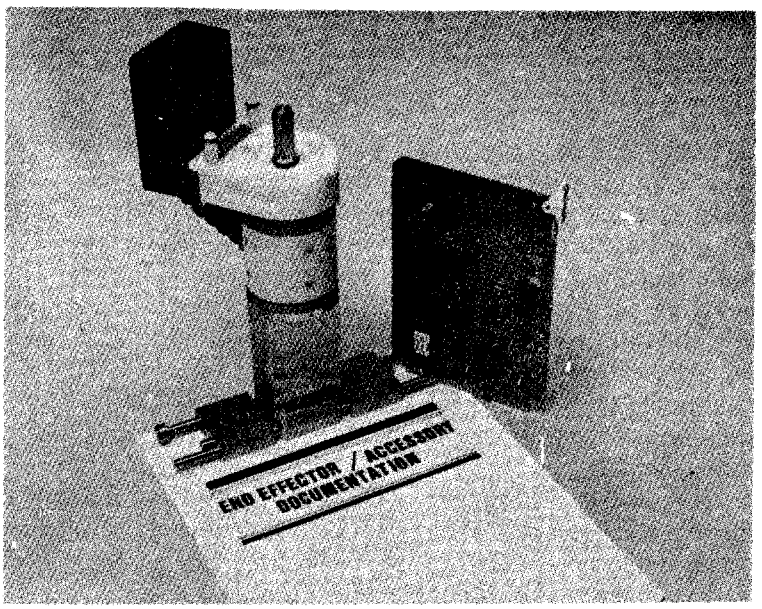


Figure 9-22 A parallel-jaw gripper using a rack-and-pinion (Intelledex 660T). (Courtesy of Intelledex, Inc., Corvallis, OR.)

The tool-tip position of the rack-and-pinion gripper remains fixed during opening and closing operations. However, the width of the rack-and-pinion gripper is typically larger than that of a parallel bar gripper, and a wide gripper may not be suitable for manipulating tall, closely spaced parts.

Recall from Chap. 4 that parts are often grasped by using an approach vector r^3 that is orthogonal to the surface on which they are resting. When this technique is feasible, the only other degree of freedom left is the roll angle q_n about the approach vector. There is no general rule for selecting q_n that is guaranteed to succeed in all circumstances. However, if simple polygonal parts, such as those shown in Fig. 9-23, are being grasped from above, then a technique that is often effective is to choose q_n to align at least one fingertip surface with a vertical face of the part, preferably the longest face.

Notice that in the case of a regular polygon, both fingertip surfaces will contact a face if the number of vertices is even, whereas one will contact a face and the other an edge if the number of vertices is odd. For elongated parts such as the rectangle, choosing the longest face increases the likelihood that the fingers will be

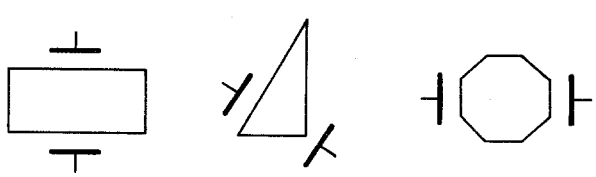


Figure 9-23 Grasping simple polygonal parts from above.

able to open sufficiently wide to grasp the part. When there is no opposing face parallel to the longest face, as in the case of the triangle, then the position of the tool tip should be selected so the other fingertip surface contacts an opposing edge. If the polygon is irregular, the tool-tip position p may no longer coincide with the centroid of the part. Therefore, when the part is transported, gravity may generate moments which cause the part to rotate within the gripper. These moments will have to be counteracted through the friction generated by a sufficiently strong gripping force.

When the tool is not a parallel-jaw gripper, the grasp-planning problem can become much more complex. There are a wide variety of standard and custom tools ranging from multifingered articulated hands (Salisbury and Craig, 1982; Jacobsen et al., 1984) to vacuum pencils and magnetic pickups. The grasping strategies for these tools depend as much on the nature of the part being grasped and the planned manipulation of it as on the physical characteristics of the tool itself.

9-6 FINE-MOTION PLANNING

If the task to be performed involves some form of assembly operation where parts have to be mated, then *fine-motion planning* techniques must be used. Fine motions are motions in which the mobile part comes into physical contact with the environment. Uncertainty plays a crucial role in fine-motion planning. Given the uncertainty in the position, orientation, size, and shape of parts, there is no guarantee that suitable contact can be achieved using pure position control based on nominal values of the part variables. Instead, the planned motion must use *feedback* from sensors to achieve contact yet avoid excessive reaction forces which may disturb the arrangement of parts. More generally, natural constraints imposed by contact with the environment can be used to *guide* the motion of the robot; for example, the mobile part might be made to slide along a surface of a fixed part.

There are a number of approaches to fine-motion planning including the impedance control technique discussed in Chap. 7. In this section we examine an alternative approach based on the use of a *generalized damper* (Lozano-Perez, et al., 1984):

$$F^{\text{tool}} = B(v - v^0) \tag{9-6-1}$$

Here F^{tool} represents the end-of-arm force and moment vector, v^0 is a unit vector representing the *command velocity* of the tool, and v is a unit vector representing the *actual velocity* of the tool. The 6×6 matrix B is the *damping matrix*. For simplicity, we assume that B is diagonal, with $B = bI$ for some damping factor $b > 0$. If the tool and the mobile part it is transporting are moving in free space, then $v = v^0$, in which case there is no end-of-arm force and moment vector F^{tool}. However, when the mobile part comes into contact with the environment, $v \neq v^0$ and reaction forces and moments are induced at the tool as a result of the contact. In this case, the control system which implements the generalized damper will attempt to maintain an actual tool velocity v that is as close to the command velocity v^0 as possible subject to the physical constraints imposed by contact with the environment. For example, the component of the velocity v orthogonal to the contact surface will be zero, but there may be a nonzero component of v tangent to the contact surface.

The degree to which the actual velocity approximates the command velocity when parts are in contact depends on the frictional characteristics of the surfaces. The reaction force due to contact at a point on the surface will lie within a *friction cone* whose axis is normal to the surface, as shown in Fig. 9-24. The angle ψ representing the radius of the friction cone is called the *friction angle*. The friction angle can be computed from the coefficient of friction μ as follows:

$$\psi = \text{atan2}\ (\mu,\ 1) \tag{9-6-2}$$

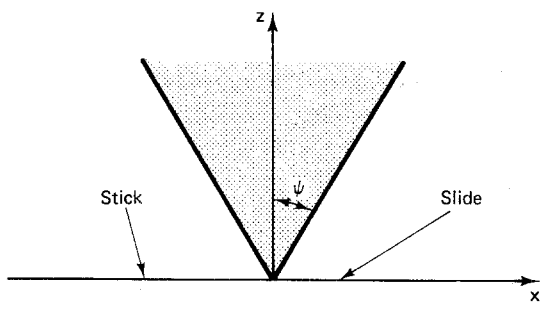


Figure 9-24 A friction cone.

Depending on the angle of the command velocity v^0, the mobile part will either slide or stick when it strikes the surface. If the angle between v^0 and the surface is sufficiently small that v^0 lies outside the friction cone, then the mobile part will *slide* along the surface, since the component of v^0 tangent to the surface is sufficiently large to overcome friction. However, if the direction of v^0 lies inside the friction cone, then the mobile part will *stick* when it contacts the surface.

9-6-1 Guarded Motion

To insert a mobile part into an assembly of fixed parts, the mobile part must be brought into contact with the fixed parts. As an illustration of this basic operation, consider the arrangement of two parts shown in Fig. 9-25. Here the objective is to move part A until vertex p comes into contact with surface G of part B. We refer to surface G as the *goal* surface.

Given the uncertainty in the part configurations, we cannot rely on nominal position control to move p to surface G. Instead, we must move p in the *direction* of G and sense when a reaction force arises. Moving in a specified direction until an

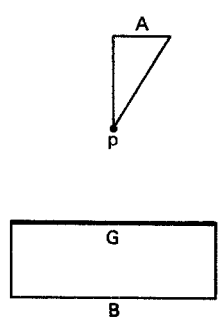


Figure 9-25 Contacting goal surface G with vertex p.

event occurs is referred to as a *guarded move*. The guard, or termination predicate, in this case is the arrival at G, which is indicated by the condition $\|F^{\text{tool}}\| > \epsilon$ for some force and moment threshold $\epsilon > 0$. To guarantee that the guarded move will successfully reach the goal surface G, we must choose an appropriate value for the nominal velocity v^{nominal} The command velocity v^0 consists of a nominal value plus an error term which represents uncertainty:

$$v^0 = v^{\text{nominal}} + \Delta v \tag{9-6-3}$$

$$\|\Delta v\| < \Delta v^{\max} \tag{9-6-4}$$

Here v^{nominal} is a unit vector which specifies a direction, while Δv is a small error vector which specifies the difference between the nominal command velocity v^{nominal} and the actual command velocity v^0, as illustrated in Fig. 9-26.

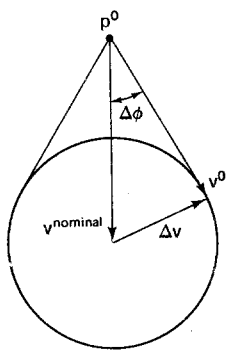


Figure 9-26 Uncertainty in the direction of v^0.

Note that the end of v^0 lies within a ball of uncertainty of radius $\Delta v^{\max}$, where $\Delta v^{\max}$ is the maximum length of the error vector Δv. It follows from Fig. 9-26 that v^0 will lie within a cone with axis v^{nominal} and angular radius $\Delta\phi$ where:

$$\Delta\phi = \arcsin(\Delta v^{\max}) \tag{9-6-5}$$

If $\Delta v^{\max}$ is sufficiently small, then $\Delta\phi \approx \Delta v^{\max}$. To determine the set of initial positions p^0 for which the guarded move will successfully strike the goal surface in spite of the error Δv in the command velocity, we compute the *preimage* of the goal surface G associated with v^0 (Lozano-Perez et al., 1984). For example, if the nominal velocity is orthogonal to the goal surface and pointing down, then the preimage, Pre (G, v^0), is an inverted cone, as shown in Fig. 9-27.

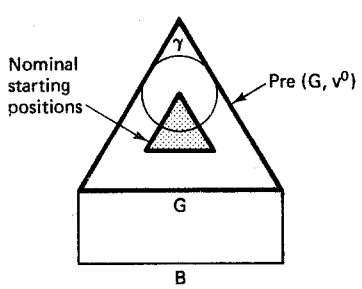


Figure 9-27 Preimage of goal surface G associated with v^0.

Any initial position p^0 starting within the set Pre (G, v^0) is guaranteed to generate a trajectory which contacts and sticks on the goal surface G. The angle γ is computed from the direction error $\Delta\phi$ and the friction cone angle ψ as follows:

$$\gamma = 2 \min \{\Delta\phi, \psi\} \tag{9-6-6}$$

Here the first factor in Eq. (9-6-6) represents a bound on the variation between the direction of v^0 and the direction of v^{nominal}, while the second factor guarantees that when contact is made, the part will stick rather than slide, because v^0 is within the friction cone.

To plan a guarded move to contact the goal surface, we must also specify a nominal value for the *initial position* vector p^0. The initial position of vertex p of the mobile part again includes a nominal value plus an error term which represents uncertainty:

$$p^0 = p^{\text{nominal}} + \Delta p \tag{9-6-7}$$

$$\| \Delta p \| < \Delta p^{\max} \tag{9-6-8}$$

Thus the starting position p^0 is known only within some ball of uncertainty of radius $\Delta p^{\max}$ centered at p^{nominal}. The nominal starting position must be restricted to a subset of the preimage Pre (G, v^0), as shown in Fig. 9-27. This way, in spite of uncertainties in both the starting position p^0 and the command velocity v^0, the following trajectory $p(t)$ is guaranteed to contact and stick on the goal surface:

$$p(t) = p^0 + \epsilon v^0 t \qquad t \geq 0 \tag{9-6-9}$$

Since we do not know the precise time t at which contact will occur, the guarded move must be performed *slowly;* that is, the parameter ϵ in Eq. (9-6-9) must be sufficiently small. This way the reaction force F^{tool} at the end of the arm can be continuously monitored so that the move can be halted as soon as contact is detected.

The guarded move specified in Eq. (9-6-9) is an example of a *guarded translation* of the mobile part. If the goal is to bring a face of the mobile part into contact with surface G of the fixed part, then a two-step procedure can be used. First the mobile part is rotated, as needed, and then a guarded translation is performed to bring a vertex p of the mobile part into contact with goal surface G of the fixed part. Once contact is achieved, a *guarded rotation* of the mobile part about the point of contact is then performed until the two surfaces come into contact. Surface contact is detected by the generation of a reaction moment about the point of rotation.

The end-of-arm force and moment vector can be measured, at least in principle, by monitoring the torques and forces induced at the joints using the Jacobian-based methods that are discussed in Chap. 5. This approach is most effective for a direct-drive robot, where there are no intervening gear reduction mechanisms. Alternatively, a force and moment sensor can be mounted at the wrist between the end of the forearm and the tool. A picture of a commercial six-axis force and moment sensor, the Lord Corporation Model 30/100, is shown in Fig. 9-28. This particular unit is capable of detecting up to 30 lb of force with a resolution of 1.152 oz and up to 100 in.-lb of torque with a resolution of 0.781 in.-oz.

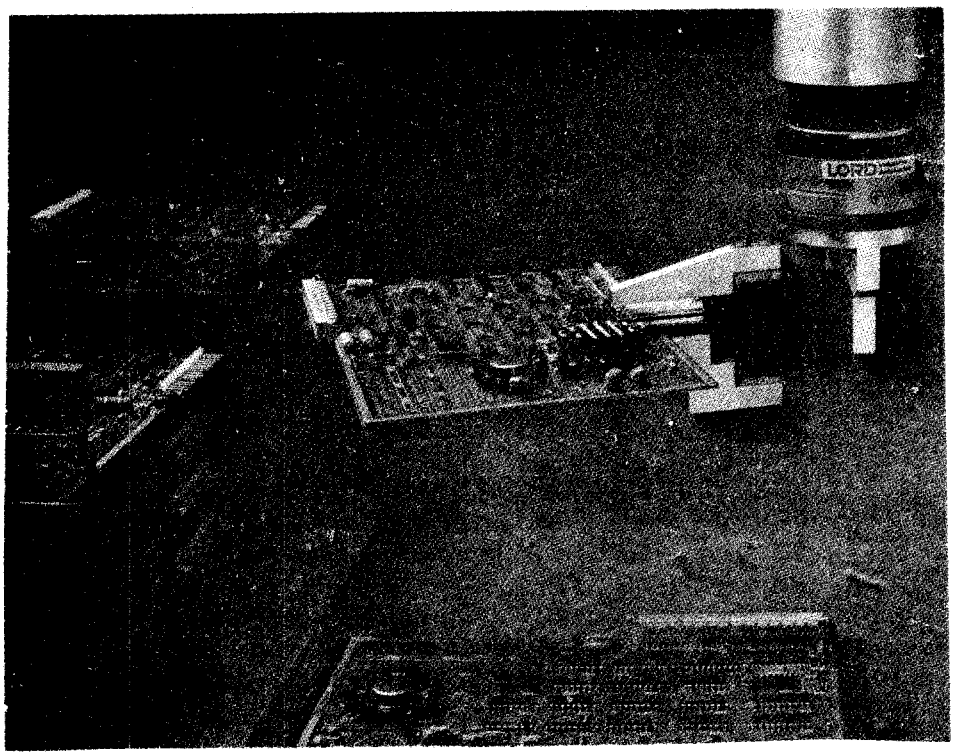


Figure 9-28 An end-of-arm force and moment sensor. (Courtesy of Lord Corporation, Industrial Automation Division, Cary, NC.)

Although guarded motions are typically used to initiate contact with the environment, other events can also be used to guard moves. For example, when the robot is reset to its hard home position, this is a guarded move in joint space. Here the termination condition for each axis is the closing of a microswitch which indicates that the axis is at its hard home position.

9-6-2 Compliant Motion

After the initial contact between a mobile part and a fixed part has been achieved through the use of a guarded move, there are often additional motions needed to obtain the desired final configuration of the part. This is the case when several faces of the mobile part must be brought into simultaneous contact with surfaces of fixed parts. After initial contact is achieved, subsequent motions can be implemented by sliding along a surface of the fixed part until contact with additional surfaces is achieved. Since this type of motion must comply with the constraints imposed by the sliding surface, these movements are referred to as examples of *compliant motion*. For compliant motion, continuous contact with the environment is a natural constraint that is used to *guide* the movement of the mobile part.

To illustrate the use of compliant motion, consider the arrangement of two parts shown in Fig. 9-29. Here the objective is to place the mobile part A in the corner of fixed part B with faces F_1 and F_2 of part A in contact with surfaces G_1 and G_2 of part B, respectively. One approach to this problem is to rotate part A as needed and then perform a guarded translation until the vertex between faces F_1 and F_2

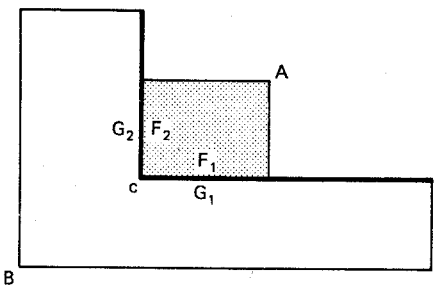


Figure 9-29 Placing a block in a corner.

strikes goal surface G_1 at some point safely to the right of the corner point c. A guarded clockwise rotation about the point of contact can then be performed until face F_1 is in contact with surface G_1. It then remains to move part A to the left until face F_2 is in contact with surface G_2.

If we pick a nominal value for v^0 that is orthogonal to G_2 and compute the preimage Pre (G_2, v^0), we can then perform a guarded move that ensures contact between F_2 and G_2. The problem with this approach is that, in the process of making the guarded move, the previously achieved contact between face F_1 and surface G_1 may be lost. Therefore a sequence of guarded moves is insufficient to guarantee the successful completion of this task. Instead we must use a compliant move in which surface G_1 is used to guide the motion. Since p^0 is already determined, it remains to choose a nominal value for v^0 which will maintain contact with surface G_1 yet still allow the mobile part to slide over to surface G_2. In this case, the direction of the unit vector v^{nominal} must lie within the tilted inner cone shown in Fig. 9-30. Here the outer cone of size $\pi/2 - \psi$ represents the set of command velocities which guarantee that part A slides to the left while maintaining contact with surface G_1. An inner cone of size $\pi/2 - \psi - 2\Delta\phi$ represents the set of nominal values for the command velocity which guarantee that the actual command velocity remains within the outer cone.

In implementing the sliding motion of A along surface G_1, the generalized

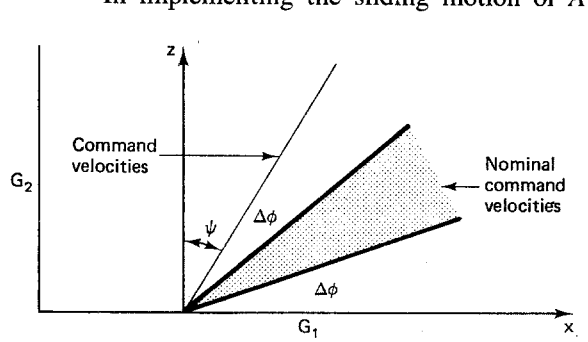


Figure 9-30 Command velocities for compliant motion along G_1.

damper controller maintains a small positive reaction force normal to the sliding surface. While this is being done, the reaction force tangent to the sliding surface must be continuously monitored, because it will increase abruptly when contact with goal surface G_2 is achieved. Thus a guarded complaint move is implemented along surface G_1, the termination predicate being contact with G_2.

Placing a block in a corner is a special case of a more difficult problem, the problem of placing a peg in a hole. The peg-in-hole problem is an important task because it can be used to model insertion and fastening operations. The peg-in-hole problem has received considerable attention in the literature, and a variety of solutions including tilting the peg and the use of chamfers have been proposed. Active compliance techniques such as those introduced here can also be used for the peg-in-hole insertion problem. In addition, a *passive* compliance device called a remote center compliance (RCC) unit has been developed and has been shown to be effective (Whitney, 1982). Figure 9-31 is a picture of a commercial RCC device, the Lord Corporation Model RCC77C01. This device allows for a lateral misalignment between the peg and the hole of up to ± 0.20 in. and angular misalignment of up to ± 1.4 degrees.

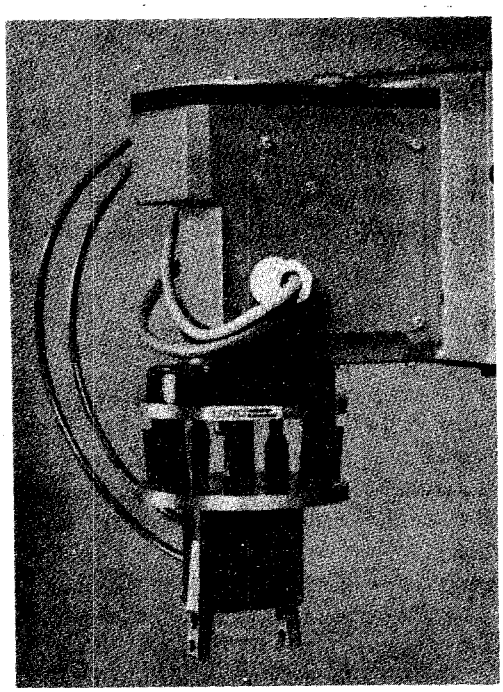


Figure 9-31 A remote center compliance (RCC) Device. (Courtesy of Lord Corporation, Industrial Automation Division, Cary, NC.)

With a passive RCC device, the wrist is sufficiently *pliable* that the constraints generated by the surfaces of the hole can tilt the peg enough to avoid jamming and wedging caused by misalignment. Passive RCC-type devices can be quite effective, but they are applicable only to specialized tasks, whereas the active compliance methods are more widely applicable.

9-7 SIMULATION OF PLANAR MOTION

Both gross-motion planning and fine-motion planning involve the concept of collisions between parts. For gross-motion planning, the planned path must be *validated* to make sure that collisions are avoided. In the case of fine motion, trajectories are designed with the objective of *causing* specific carefully controlled collisions to occur. To evaluate a planned motion, a simulation of it should be performed. We examine some simple simulation techniques by restricting our attention to the motion of convex polygons within a plane.

Consider the following representation of a line in the xy plane:

$$ax + by + c = 0 \tag{9-7-1}$$

Unlike the familiar slope-intercept formulation of a line, Eq. (9-7-1) is a general formulation which represents all types of straight lines, including those with infinite slope. If (x_1, y_1) and (x_2, y_2) denote the coordinates of two distinct points on the line, then the parameters of the line through these points can be computed as follows:

$$a = y_1 - y_2 \tag{9-7-2a}$$

$$b = x_2 - x_1 \tag{9-7-2b}$$

$$c = x_1 y_2 - x_2 y_1 \tag{9-7-2c}$$

To detect collisions between polygons, first we must develop a measure of the distance from a point to a line. Given a line represented by the parameters $\{a, b, c\}$ and a point with coordinates (x_0, y_0), the minimum Euclidean distance from the point to the line is:

$$d(x_0, y_0) \triangleq \frac{ax_0 + by_0 + c}{(a^2 + b^2)^{1/2}} \tag{9-7-3}$$

Note that if (x_0, y_0) is on the line, then from Eq. (9-7-1) we see that $d(x_0, y_0) = 0$. The distance function d is a signed quantity, with the sign of d indicating which side of the line the point is on and the magnitude of d representing the distance between the point and the line. Given a line from point (x_1, y_1) to point (x_2, y_2), the value of d is positive when (x_0, y_0) is to the left of the line, zero when it is on the line, and negative when it is to the right of the line.

The formulation in Eq. (9-7-3) can now be used to determine the amount by which a point penetrates a convex polygon. We represent a polygon P as an ordered list of vertices, as follows:

$$P = \{(x_k, y_k): 1 \leq k \leq n + 1\} \tag{9-7-4}$$

Here it is assumed that the n vertices of P are ordered in a counterclockwise sense about the centroid. We can simplify subsequent processing by augmenting the list with an additional vertex, $(x_{n+1}, y_{n+1}) = (x_1, y_1)$, thereby making the list circular. To detect whether or not point (x_0, y_0) lies inside polygon P, and to determine the amount of penetration, we successively compute the distance function $d(x_0, y_0)$ using the lines generated by the edges of P. The point (x_0, y_0) lies inside P if and only if

the n distances $\{d_k(x_0, y_0)\}$ are all positive. If point (x_0, y_0) is inside P, the amount of penetration is equal to the minimum value of the n distances. In particular, we have the following algorithm, in which P is assumed to be a convex polygon with vertices ordered in a counterclockwise sense.

Algorithm 9-7-1: Polygon Penetration

0. Set $k = 1$, $d_0 =$ maximum real.

1. Compute:

$$a = y_k - y_{k+1}$$

$$b = x_{k+1} - x_k$$

$$c = x_k y_{k+1} - x_{k+1} y_k$$

$$d = \frac{ax_0 + by_0 + c}{(a^2 + b^2)^{1/2}}$$

2. If $d < d_0$, then set $d_0 = d$.

3. Set $k = k + 1$. If $k \leq n$, go to step 1.

The output of Algorithm 9-7-1 is the *penetration distance* d_0. Point (x_0, y_0) lies inside P if and only if $d_0 > 0$. In this case d_0 specifies the amount of penetration, as shown in Fig. 9-32. When $d_0 = 0$, point (x_0, y_0) lies on the boundary of polygon P.

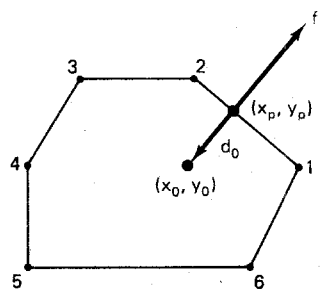


Figure 9-32 Measuring penetration of a convex polygon P.

It is important to note that Algorithm 9-7-1 is valid only for *convex* polygons, polygons for which the interior angles are each less than or equal to π. This is why we work exclusively with convex polygons. Recall that there is no loss of generality, because nonconvex polygons can always be regarded as unions of convex polygons.

If the only objective is to determine whether or not point (x_0, y_0) penetrates polygon P, then Algorithm 9-7-1 can be made more efficient. For example, as soon as d in step 1 becomes negative, the procedure can be terminated, because (x_0, y_0) does not lie inside polygon P in this case. Furthermore, in evaluating d, there is no need to normalize with $(a^2 + b^2)^{1/2}$, because only the sign of d is of interest. Another way to make Algorithm 9-7-1 more efficient is to *precompute* the parameters of the edges of P and thereby develop the following alternative representation of the polygon:

$$P = \{(a_k, b_k, c_k): 1 \leq k \leq n + 1\} \tag{9-7-5}$$

Algorithm 9-7-1 can be used to simulate a force sensor for detecting a collision between a vertex and a polygonal surface during a guarded move. In this case the amount of penetration will determine the *magnitude* of the reaction force, and the line from (x_0, y_0) to the penetration point (x_p, y_p) in Fig. 9-32 will determine the *direction* of the reaction force. If the edge being penetrated has vertices (x_1, y_1) and (x_2, y_2), then the x and y components of the reaction force can be computed as follows:

$$F_1^{\text{tool}} = \alpha[x_1 - x_0 + \lambda(x_2 - x_1)]1(d_0) \tag{9-7-6a}$$

$$F_2^{\text{tool}} = \alpha[y_1 - y_0 + \lambda(y_2 - y_1)]1(d_0) \tag{9-7-6b}$$

$$\lambda = \frac{(x_0 - x_1)(x_2 - x_1) + (y_0 - y_1)(y_2 - y_1)}{[(x_2 - x_1)^2 + (y_2 - y_1)^2]^{1/2}} \tag{9-7-6c}$$

Here $1(d_0)$ denotes the unit step function, which is included to ensure that there is no reaction force unless and until penetration of P occurs. To derive F^{tool}, the point of penetration (x_p, y_p) is first computed. This is done by translating the origin to (x_1, y_1) and then computing the orthogonal projection of (x_0, y_0) onto the line or subspace generated by (x_2, y_2). The vector from (x_0, y_0) to (x_p, y_p) is then scaled by a constant $\alpha > 0$ to produce F^{tool}. The relationship between the amount of penetration d_0 and the magnitude of the reaction force is shown in Fig. 9-33. Notice that a springlike mechanism is assumed for the surface of the polygon. The degree of stiffness is controlled by the spring constant α.

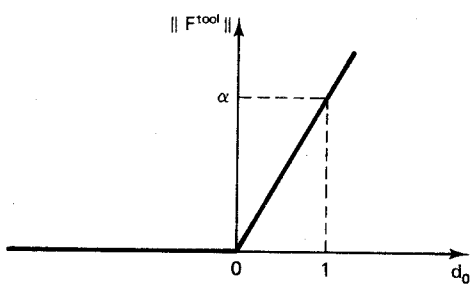


Figure 9-33 A simulated force sensor.

In using Eq. (9-7-6) to model a force sensor for simulation purposes, the exact values for the part positions can be used to detect collisions even though the paths must be planned using only the nominal values plus feedback from the force sensor. The simulated force sensor in Eq. (9-7-6) can be used for both guarded translations and guarded rotations. In the case of guarded rotations, the moment about the point of rotation can be computed once the reaction force and point of penetration are known.

Exercise 9-7-1: Polygonal Collision. Write an algorithm which returns the distance by which convex polygon A penetrates convex polygon B. *Hint:* You can use Algorithm 9-7-1 as a step in your algorithm.

9-8 A TASK-PLANNING PROBLEM

Recall that task-level programming is a high-level technique for controlling a manipulator that requires the user to supply the goals of the manipulation task rather than specific means used to achieve these goals. An effective way to specify simple goals is through the use of robot vision.

9-8-1 Source and Goal Scenes

The basic assumption is that data is made available to the task planner in the form of an ordered pair of scenes or images, a *source scene* and a *goal scene*. The source scene S specifies the initial layout of parts in the workspace, while the goal scene G specifies the final layout or desired rearrangement of parts in the workspace. The objective is to use the information contained in the two scenes, plus feedback from force and moment sensors, to plan and implement a sequence of motions which cause the robot to redistribute the parts in the source scene so as to *correspond* to the goal scene. If the task-planning algorithm is represented by an operator Manip acting on the source scene S, then the task-planning operation can be represented symbolically as follows:

$$G \subseteq \text{Manip } (S) \tag{9-8-1}$$

Notice that the goal scene G is a subset of the manipulated source scene Manip (S). In many instances, G = Manip (s), but this is not necessarily true, because the source scene may contain redundant parts which do not appear in the goal scene. A simple example showing a source scene and a goal scene is displayed in Fig. 9-34. Here the source scene consists of four convex polygons in a plane, three of which are identical copies of one another. In the goal scene the parts are rearranged to form a simple houselike structure. Since there are no redundant parts in S, the goal scene is identical to the manipulated source scene in this case.

The example illustrated in Fig. 9-34 is a task-level programming problem because a goal is specified, but not an explicit procedure for reaching that goal. There are many potential applications that fall within the {S, G, Manip} paradigm. For example, the source scene might correspond to parts in a bin, parts in a part feeder, parts on a pallet, or parts on a conveyor belt. The goal scene could then correspond to one or more parts inserted into a subassembly, or a part loaded into a machine or processing station. In instances like Fig. 9-34 where G = Manip (S), the source and goal scenes can be interchanged, thus *reversing* the task. Thus parts can be unloaded rather than loaded, extracted rather than inserted; and objects can be disassembled rather than assembled. Of course, certain physical operations are not readily reversible, such as those that depend upon gravity or those that use special part fixturing devices.

The most interesting class of applications lies in the area of assembly. If the objective is to assemble an object from numerous basic parts, then a *sequence* of goal scenes can be used corresponding to a series of increasingly complex subassemblies, with the final goal scene representing the completed assembly. Note that this

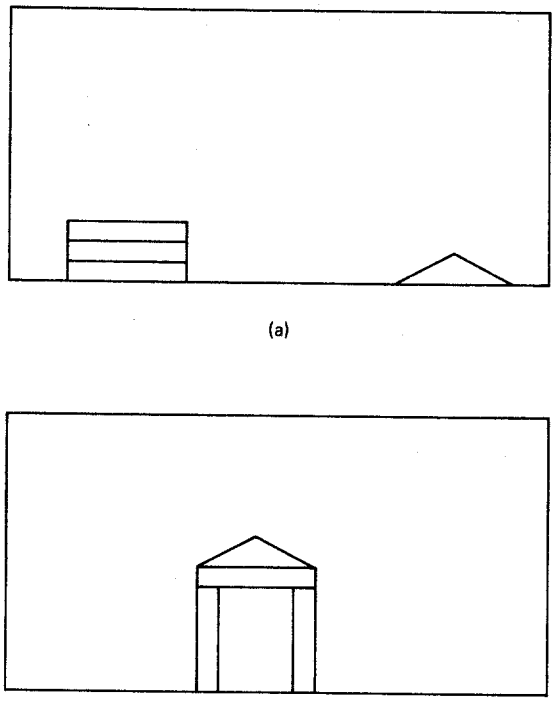


Figure 9-34 A source scene (a) and a goal scene (b).

approach to assembly is not unlike instructions given to humans to assemble simple mechanical kits. Assembly instructions for humans typically include written instructions or steps which accompany the sequence of views of the partially assembled object. An exploded view showing all the individual parts corresponds to the original source scene, while views of the partially completed subassemblies correspond to a series of goal scenes.

The objective, then, is to develop a high-level robot motion planning algorithm for automated manipulation based on the use of an ordered pair of scenes obtained from a vision system. Although this problem is conceptually simple to pose, its solution involves a number of subtle notions. The motivation for this high-level approach lies in the observation that it has potential for significant payback in industrial applications. It simplifies the human effort needed to program a robotic manipulator, and it also reduces the need for high-precision part presentation devices.

9-8-2 Task-Planning Subproblems

We make no attempt to present a general solution to the task-planning problem we have posed, because the solution is still a topic of research. However, we do identify important subproblems which must be addressed if a functional task planner is to be implemented. These task-planning subproblems are listed in Table 9-4. The last three of these subproblems, gross-motion planning, grasp planning, and fine-motion

TABLE 9-4 TASK-PLANNING SUBPROBLEMS

Subproblem	Description
1	Scene analysis
2	Part ordering
3	Gross-motion planning
4	Grasp planning
5	Fine-motion planning

planning, have been discussed previously. We therefore focus on the remaining two subproblems.

Scene analysis. The *scene analysis* subproblem is one of processing the two scenes $\{S, G\}$ with the objective of associating with each part in the goal scene G one or more corresponding parts in the source scene S. The simplest special case occurs when both the source scene and the goal scene consist of *disjoint* collections of parts. Here every part in the goal scene appears somewhere in the source scene, although at a random position and orientation. For example, the source scene might consist of parts on a conveyor belt near a processing station, and the goal scene could consist of a part placed in the processing station.

The assumption that the parts are disjoint is overly restrictive, because in practical cases parts may be *partially occluded* because of overlap. Here *local* features of the parts must be used to identify and locate them (Turney et al., 1985). Partial occlusion will occur, for example, if the parts are available from a bin in the source scene or if the parts are assembled or mated in the goal scene. The problem of partially or even totally occluded parts in the source scene can be addressed by using the robot as an *active* device to reduce uncertainty in the source scene. In this case, if a part in the goal scene cannot be successfully identified in the source scene, then the parts in the source scene can be *redistributed* by the robot to produce a new, more revealing source scene. The scene analysis process can then be resumed. These steps can be applied recursively until the part in the goal scene is successfully identified in the source scene. For example, if the goal is to extract a part from a stack of parts and the desired part is not visible, then the occluding parts can first be removed from the stack.

While the parts in the source scene can be isolated and identified through active intervention with the robot, this approach is not feasible for partially occluded parts in the goal scene. Indeed, this is a key problem that goes to the heart of the assembly operation. If two parts are to be *mated* in the goal scene, then partial occlusion may be inevitable. For example, the mating of parts might take the form of a press fit, which is an instance of the peg in-hole problem (Whitney, 1982). Other examples of common fasteners which cause partial occlusion are a nuts, bolts, and screws. The problem of a totally occluded or *hidden* part in the goal scene can be eliminated by using an appropriate *series* of goal scenes.

Part ordering. Another important subproblem is the *part ordering* subproblem (Sacerdoti, 1977; Popplestone et al., 1980). Here the problem is to determine

the *order* in which parts in the source scene are to be moved in order to achieve the desired arrangement of parts specified in the goal scene. The simplest way to solve this subproblem is to have the programmer do it by specifying a detailed *sequence* of goal scenes in which only one part changes configuration between successive goal scenes. This is a degenerate version of the task-planning problem which effectively removes the part ordering responsibility from the task planner. When several parts differ between the source and goal scenes, then the order in which parts are to be moved must be determined.

There are several issues that must be addressed in this case. To begin with, the arrangement of parts in the goal scene may be such that one or more parts in the source scene first have to be moved or *cleared* away to make room for the goal scene arrangement. This is an instance of the *findspace* problem (Lozano-Perez, 1983b). The decision on where to put the offending part or parts is not entirely arbitrary, because these parts then become obstacles for the subsequent gross-motion subproblem.

Once space has been cleared to construct the goal scene, the order in which the parts in the goal scene are to be put in place has to be determined. Although there may be considerable flexibility at this point, the order is by no means arbitrary. For example, the existence of gravity dictates that if part B is supported by part A, then part A has to be put into place before part B. Similarly, a part which is totally enclosed by other parts has to be put into position before the enclosure is complete. Subject to these and similar constraints, one might attempt to order the movement of parts so as to minimize or at least reduce some performance index such as the total distance moved or perhaps the number of movements.

The order in which the parts in the goal scene are to be put in place can also be influenced by the arrangement of parts in the source scene. For example, if an analysis of the goal scene dictates that a particular part should be put into place next, and if this part is at the bottom of a stack of parts in the source scene, then all the other parts will have to be moved to intermediate locations first, another instance of the findspace problem. When there are multiple copies of a part in the source scene, the question of which one is best to use must also be addressed using some performance criterion.

The part ordering subproblem is a key question because it can affect the difficulty and ultimately the success of rearranging the parts in the source scene to achieve the goal scene. Recall that instructions given to humans to assemble kits already provide this ordering information by supplying a series of written steps to follow. The task planner is not provided this information, and must therefore synthesize a workable sequence on its own. Just how divergent the source and goal scenes can be without making the part ordering problem intractable is an open question.

9-8-3 Task Planner Simulation

To illustrate the notion of task planning, we simulate a very rudimentary task planner, one applicable to planar motion of disjoint collections of convex polygonal parts. For gross-motion planning, we use the configuration-space method to shrink the mobile part to a point while growing the polygonal obstacles to compensate.

Since the shapes of the grown obstacles vary as the mobile part is rotated, we first compute a fixed *traveling* orientation which is selected to minimize a weighted measure of the overlap between grown obstacles. Points in free space are then found where rotations to and from the traveling orientation can be safely performed. Once these start and stop points are determined, the shortest collision-free path between these points is computed by searching the graph of visible obstacle vertices (Lozano-Perez and Wesley, 1979). This technique is then used as a step in a higher-level algorithm, an algorithm to plan a sequence of moves to rearrange a source scene to correspond to a goal scene.

A simple illustration of this technique is shown in Fig. 9-35. For this example, a cross-reference table identifying corresponding parts in the two scenes was supplied directly to the task planner, thereby circumventing the scene analysis subproblem. A rudimentary solution to the part ordering subproblem was implemented on the basis of two heuristic rules. The first rule was that parts in the goal scene were put into place by proceeding from the boundary of the workspace toward the center. The second rule resolved the question of how to choose between identical copies of a

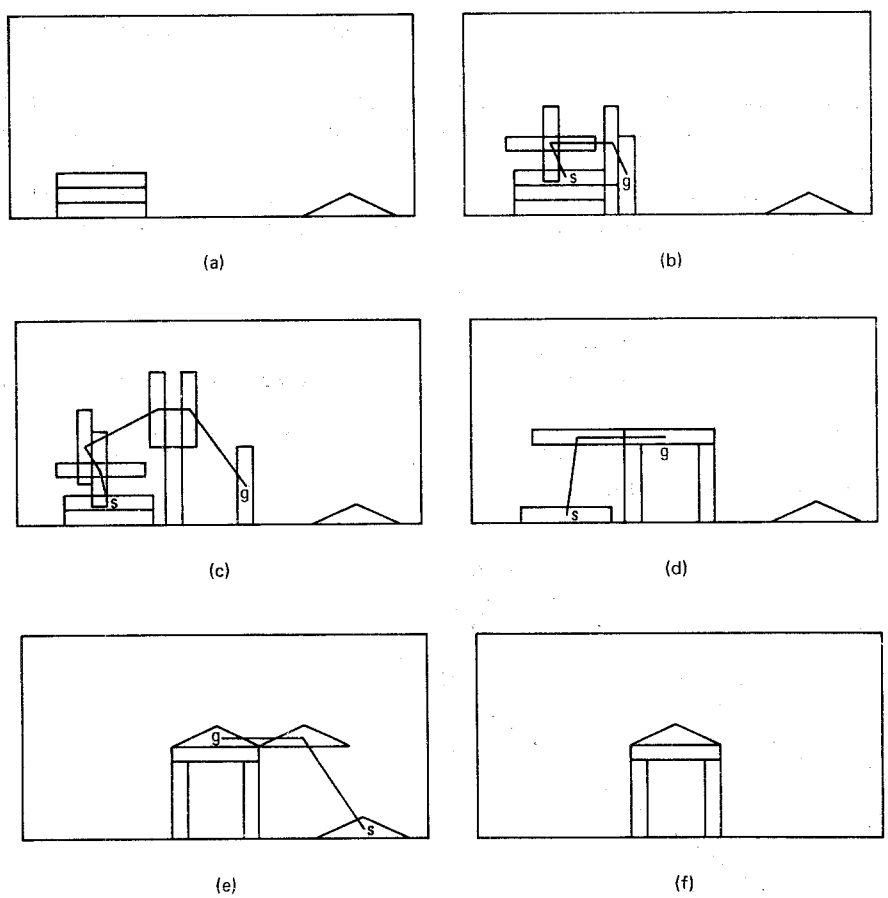


Figure 9-35 A simple task-planning example.

part in the source scene. Here the part farthest from the workspace boundary was chosen first. There was no grasp planning or fine-motion planning implemented in this simple example. Instead the simulated parts were moved directly, and exact values for the part positions and orientations were assumed.

9-9 PROBLEMS

9-1. Consider the scene shown in Fig. 9-36. Suppose A is the mobile part and B_1 and B_2 are fixed obstacles. Sketch the configuration-space scene induced by A, using reference point r.

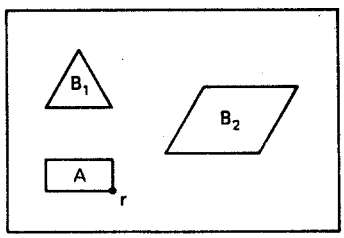


Figure 9-36 A workspace with two obstacles.

9-2. Repeat Prob. 9-1, but with the mobile part rotated by $\pi/2$. That is, sketch the configuration-space slice projection associated with a mobile part orientation of $\phi = \pi/2$.

9-3. Consider the scene shown in Fig. 9-37. Sketch the configuration-space scene induced by part A, using reference point r. You will have to represent A as a union of convex polygons.

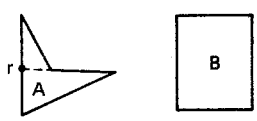


Figure 9-37 A nonconvex mobile part.

9-4. Suppose a mobile rectangle A is at position p^k and orientation ϕ_k as shown in Fig. 9-38. Find the radius r and angle θ in terms of the rectangle length L and width W. Find the coordinates of the vertices $\{a^j: 1 \leq j \leq 4\}$ in terms of p^k, ϕ_k, r, and θ.

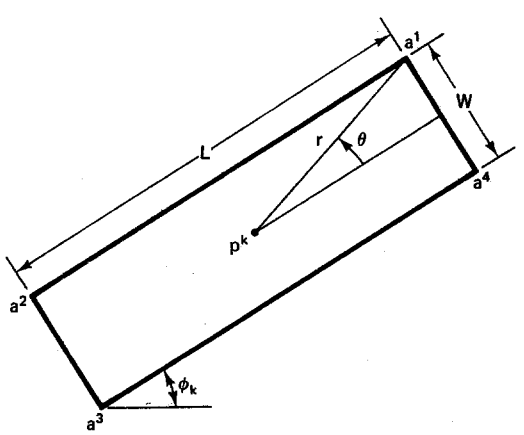


Figure 9-38 A mobile rectangle.

9-5. A gross-motion path $P = \{(p^k, \phi_k)\colon 0 \leq k \leq N\}$ has been planned for a mobile rectangle A. Here p^k denotes the position of the centroid of A and ϕ_k denotes the angle that A makes with the x axis as shown in Fig. 9-38. Write an algorithm which *validates* the path P, that is, determines whether or not collisions occur. You can assume that there are M convex polygonal obstacles $\{B_k\}$. *Hint:* Use Algorithm 9-7-1 and the solution to Prob. 9-4.

9-6. Suppose a robot is to be used to play chess. Two parallel-jaw grippers with short fingers are available. One is based on a rack-and-pinion design, while the other is based on a parallel bar design. Which gripper would appear to be most suitable for this application, and why?

9-7. Consider the two-dimensional peg-in-hole problem shown in Fig. 9-39. Sketch the preimage Pre (G, v^0) associated with a nominal command velocity angle of $\phi = 0$ which points straight down. The friction cone radius is $\psi = 35$ degrees, and the command velocity error angle is $\Delta\phi = 15$ degrees.

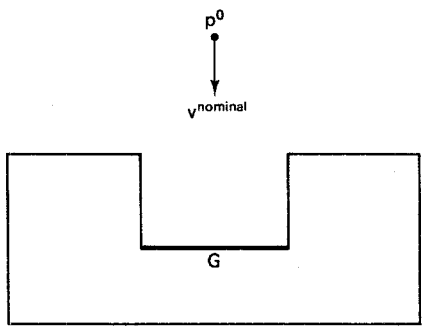


Figure 9-39 The peg-in-hole problem.

9-8. Repeat Prob. 9-7, but with a nominal command velocity angle of $\phi = 30$ degrees.

9-9. Repeat Prob. 9-7, but with a nominal command velocity angle of $\phi = -60$ degrees. Assume the hole is 2 units wide by 1 unit deep.

9-10. Consider the two-dimensional peg-in-hole problems shown in Fig. 9-40. Sketch the configuration-space scenes in each case. Does it make any difference, in configuration space, whether the chamfers are on the peg or the hole?

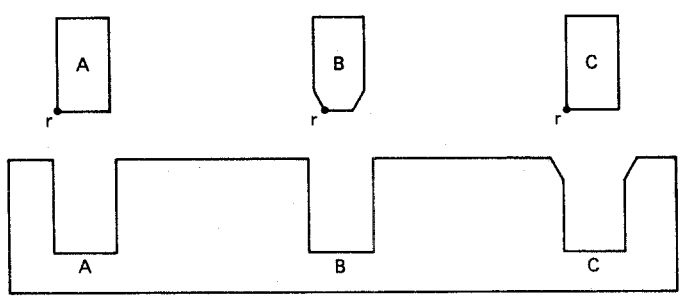


Figure 9-40 Peg-in-hole using chamfers.

9-11. Suppose peg A in Prob. 9-10 is tilted by rotating it by 30 degrees about r. Assuming the width of the peg is w, sketch the configuration space for peg A and hole A in this case.

9-12. Consider the cavity shown in Fig. 9-41. Sketch the configuration space induced by triangle A.

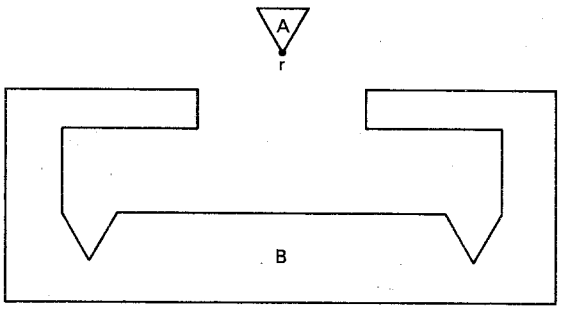


Figure 9-41 Configuration space of a cavity.

REFERENCES

BROOKS, R. A. (1982) "Symbolic error analysis and robot programming," *Int. J. Robotics Research,* Vol. 1, No. 4, pp. 29–68.

BROOKS, R. A. (1983) "Solving the find-path problem by good representation of free space," *IEEE Trans. Syst., Man, Cybern.,* Vol. SMC-13, No. 3, pp.190–197.

BROOKS, R. A., and T. LOZANO-PEREZ (1985). "A subdivision algorithm in configuration space for findpath with rotation," *IEEE Trans. Syst., Man, Cybern.,* Vol. SMC-15, No. 2, pp. 224–233.

DEISENROTH, M. P. (1985). "Robot teaching," pp. 352–365 in S. Y. Nof, ed., *Handbook of Industrial Robotics,* Wiley: New York.

GONDRAN, M., M. MINOUX, and S. VAJDA (1984). *Graphs and Algorithms,* Wiley: New York.

GRUVER, W. A., B. I. SOROKE, J. J. CRAIG, and T. L. TURNER (1983). "Evaluation of commercially available robot programming languages," *13th Int. Symp. Industrial Robot and Robots 7, Chicago,* April, Vol. 12, pp. 58–68.

JACOBSEN, S. C, J. E. WOOD, D. F.. KNUTTI, and K. B. BIGGERS (1984). "The Utah/MIT dexterous hand: Work in progress," *Int. J. Robotics Research,* Vol. 3, pp. 21–50.

LEE, D. T., and R. L. DRYSDALE III (1981). "Generalized Voronoi diagrams in the plane," *SIAM J. Comput.,* Vol. 10, No. 1, pp. 73–87.

LOZANO-PEREZ, T. (1983a). "Robot programming," *Proc. IEEE,* Vol. 71, No. 7, pp. 821–841.

LOZANO-PEREZ, T. (1983b). Spatial planning: A configuration space approach," *IEEE Trans. Computers,* Vol. C-32, pp. 108–120.

LOZANO-PEREZ, T. (1987). "A simple motion planning algorithm for general robot manipulators," *IEEE J. Robotics and Automation,* Vol. RA-3, No. 3, pp. 224–238.

LOZANO-PEREZ, T., and R. A. BROOKS (1985). "Task-level manipulator programming," *Handbook of Industrial Robotics,* pp. 404–418 in S. Y. Nof, ed., Wiley: New York.

LOZANO-PEREZ, T., M. T. MASON, and R. H. TAYLOR (1984). "Automatic synthesis of fine-motion strategies for robots," *Int. J. Robotics Research,* Vol. 3, No. 1, pp. 3–24.

LOZANO-PEREZ, T., and M. A. WESLEY (1979). "An algorithm for planning collision-free paths among polyhedral obstacles," *Comm. ACM,* Vol. 22, pp. 560–570.

O'DUNLAING, C., and C. K. YAP (1985). "A retraction method for planning the motion of a disc," *J. Algorithms,* Vol., 1, pp. 104–111.

PESHKIN, M. A., and A. C. SANDERSON (1986). "Reachable grasps on a polygon: The convex rope algorithm," *IEEE J. Robotics and Automation,* Vol. RA-2, No. 1, pp. 53–58.

POPPLESTONE, R. J., A. P. AMBLER, and I. BELLOW (1980). "An interpreter for a language for describing assemblies," *Artificial Intelligence,* Vol. 14, pp. 79–107.

SACERDOTI, E. D. (1977). *A Structure for Plans and Behavior,* American Elsevier: New York.

SALISBURY, J. K., and J. J. CRAIG (1982). "Articulated hands: Force control and kinematic issues," *Int. J. Robotics Research,* Vol. 1, p. 1.

SCHILLING, R. J., and R. WHITE (1990). *Robotic Manipulation: A Laboratory Manual,* Prentice Hall, Englewood Cliffs, N.J.

TAKAHASHI, O., and R. J. SCHILLING (1989). "Motion planning in a plane using generalized Voronoi diagrams," *IEEE Trans. Robotics and Automation* Vol. RA-5, No. 2, pp. 143–150.

TURNEY, J. L., T. N. MUDGE, and R. A. VOLZ (1985). "Recognizing partially occluded parts," *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol. PAMI-7, No. 4, pp. 410–421.

WHITNEY, D. E. (1982). "Quasi-static assembly of compliantly supported rigid parts," *J. Dynamic Systems, Measurement and Control,* Vol. 104, pp. 65–77.

WOLTER, J. D., R. A. VOLZ, and A. C. WOO (1985). "Automatic generation of gripping positions," *IEEE Trans. Syst., Man, Cybern.,* Vol. SMC-15, No. 2, pp. 204–212.

Appendix 1

Trigonometric Identities

Definitions (Fig. A-1)

$$\sin \alpha \triangleq \frac{a}{c}$$

$$\cos \alpha \triangleq \frac{b}{c}$$

$$\tan \alpha \triangleq \frac{a}{b}$$

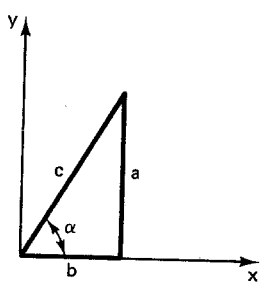


Figure A-1 Right triangle.

Pythagorean Theorem

$$a^2 + b^2 = c^2$$

$$\sin^2 \alpha + \cos^2 \alpha = 1$$

Symmetry

$$\sin(-\alpha) = -\sin\alpha$$
$$\cos(-\alpha) = \cos\alpha$$
$$\tan(-\alpha) = -\tan\alpha$$

Sums and Differences

$$\sin(\alpha + \beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$$
$$\sin(\alpha - \beta) = \sin\alpha\cos\beta - \cos\alpha\sin\beta$$
$$\cos(\alpha + \beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$$
$$\cos(\alpha - \beta) = \cos\alpha\cos\beta + \sin\alpha\sin\beta$$

Double Angle

$$\sin 2\alpha = 2\sin\alpha\cos\alpha$$
$$\cos 2\alpha = \cos^2\alpha - \sin^2\alpha$$

Half Angle

$$\sin^2\frac{\alpha}{2} = \frac{1 - \cos\alpha}{2}$$
$$\cos^2\frac{\alpha}{2} = \frac{1 + \cos\alpha}{2}$$

General Laws (Fig. A-2)

$$\alpha + \beta + \gamma = \pi$$
$$\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma}$$
$$a^2 = b^2 + c^2 - 2bc\cos\alpha$$

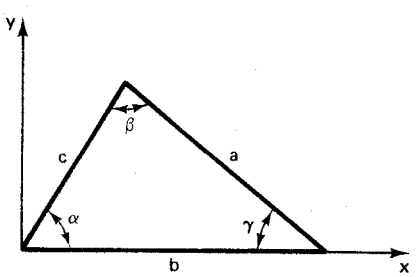


Figure A-2 General triangle.

Appendix 2

Moments of Inertia

The following moments of inertia of three-dimensional objects are based on the assumption that the objects are homogeneous with a *density* of ρ. For an object of volume V, the moments of inertia about the x, y, and z axes of an orthonormal coordinate frame are computed as follows:

$$I_{xx} = \iiint_V (y^2 + z^2)\rho \, dx \, dy \, dz$$

$$I_{yy} = \iiint_V (x^2 + z^2)\rho \, dx \, dy \, dz$$

$$I_{zz} = \iiint_V (x^2 + y^2)\rho \, dx \, dy \, dz$$

Homogeneous objects with regular shapes have moments of inertia that can be expressed in terms of the *total mass* $m = \rho V$. Figures A-3 through A-8 show some of the more common geometrical shapes. The objects shown in Figs. A-3 through A-8 can be used as building blocks to model the links of a robotic arm. Their moments of inertia are summarized in Table A2-1.

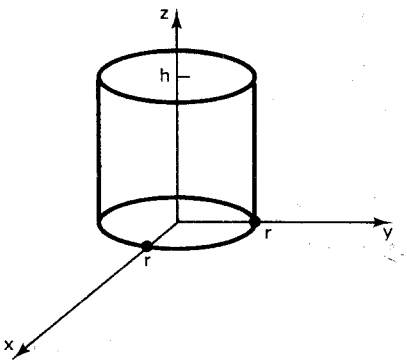


Figure A-3 Cylinder.

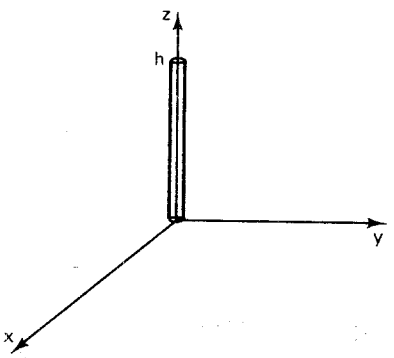


Figure A-4 Thin cylinder.

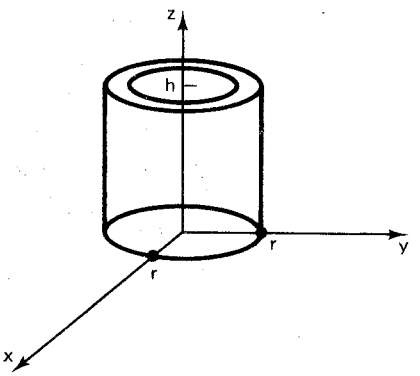


Figure A-5 Cylindrical shell.

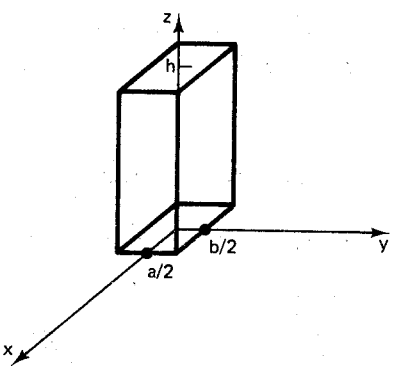


Figure A-6 Prism.

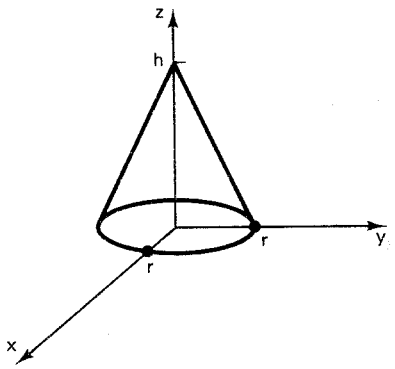


Figure A-7 Cone.

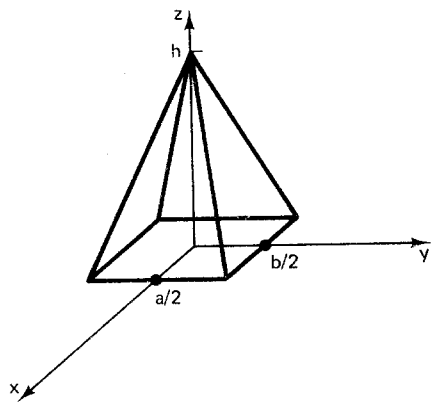


Figure A-8 Pyramid.

TABLE A2-1 MOMENTS OF INERTIA

Object	I_{xx}	I_{yy}	I_{zz}
Cylinder	$\frac{m(3r^2 + 4h^2)}{12}$	$\frac{m(3r^2 + 4h^2)}{12}$	$\frac{mr^2}{2}$
Thin cylinder	$\frac{mh^2}{3}$	$\frac{mh^2}{3}$	0
Cylindrical shell	$\frac{m(3r^2 + 2h^2)}{6}$	$\frac{m(3r^2 + 2h^2)}{6}$	mr^2
Prism	$\frac{m(b^2 + 4c^2)}{12}$	$\frac{m(a^2 + 4c^2)}{12}$	$\frac{m(a^2 + b^2)}{12}$
Cone	$\frac{m(3r^2 + 2h^2)}{20}$	$\frac{m(3r^2 + 2h^2)}{20}$	$\frac{3mr^2}{10}$
Pyramid	$\frac{m(b^2 + 2h^2)}{20}$	$\frac{m(a^2 + 2h^2)}{20}$	$\frac{m(a^2 + b^2)}{20}$

Appendix 3

List of Symbols

Given the scope and the interdisciplinary nature of the field of robotics, a large number of notational symbols must be used. The following list summarizes most of the special symbols that appear in the text. In each case the symbol is defined in the text the first time it appears.

TABLE A3-1 PARAMETERS

Symbol	Description	Dimension
t	time	1×1
t_{switch}	time to hit switching surface	1×1
T	trajectory traversal time	1×1
T_k	traversal time for segment k	1×1
ΔT	parabolic blend transition time	1×1
T_d	delay time: $y(T_d) = y(\infty)/2$	1×1
T_s	settling time	1×1
T_m	motor time constant	1×1
s	complex frequency	1×1
n	number of axes	1×1
d	joint distance vector	$n \times 1$
θ	joint angle vector	$n \times 1$
a	link length vector	$n \times 1$
α	link twist angle vector	$n \times 1$
m	link mass vector	$n \times 1$
ξ	joint type vector: 0-revolute; 1-prismatic	$n \times 1$
q	joint variable vector: $q_k = \xi_k\theta_k + (1 - \xi_k)d_k$	$n \times 1$
$\dot{q}$	joint velocity vector: dq/dt	$n \times 1$

TABLE A3-1 *(Continued)*

Symbol	Description	Dimension
$\ddot{q}$	joint acceleration vector: d^2q/dt^2	$n \times 1$
$\tilde{q}$	joint-space singularity: dex $(\tilde{q}) = 0$	$n \times 1$
$\hat{q}$	equilibrium position	$n \times 1$
$q^{\min}$	lower joint limit vector	$m \times 1$
$q^{\max}$	upper joint limit vector	$m \times 1$
x	tool-configuration vector	6×1
$\dot{x}$	tool-configuration velocity: dx/dt	6×1
p	tool position vector	3×1
$\dot{p}$	linear tool velocity: dp/dt	3×1
$\dot{\phi}$	angular tool velocity: $d\phi/dt$	3×1
τ	applied joint torque vector	$n \times 1$
τ_L	load shaft torque	1×1
$\tau_{\max}$	maximum torque	1×1
r	reference input vector	$n \times 1$
e	trajectory error vector: $e = r - q$	$n \times 1$
g	gravitational acceleration vector	3×1
g_0	gravitational constant: 9.8062 m/sec^2	1×1
F^{tool}	end-of-arm force and moment vector	6×1
f^{tool}	end-of-arm force vector	3×1
n^{tool}	end-of-arm moment vector	3×1
Δc^k	center-of-mass displacement vector for link k	4×1
b_k^v	viscous friction coefficient for joint k	1×1
b_k^d	dynamic friction coefficient for joint k	1×1
b_k^s	static friction coefficient for joint k	1×1
$\bar{D}_k$	inertia tensor of link k relative to frame L_k	3×3
σ	homogeneous coordinate scale factor	1×1
H_σ	homogeneous coordinate conversion matrix: $[I/\sigma, 0]$	3×4
η	perspective vector	3×1
b^k	intersection of x^k and z^{k-1} axes	3×1
C	joint coupling matrix	$m \times n$
ρ	screw pitch: threads per unit length, density	1×1
ν	approach distance	1×1
du	infinitesimal tool displacement vector	6×1
dp	infinitesimal tool translation	3×1
$d\phi$	infinitesimal tool rotation	3×1
dq	infinitesimal joint displacement	$n \times 1$
Δu	tool deflection vector	6×1
Δp	linear tool deflection, tool-tip precision	3×1
$\Delta\phi$	angular tool deflection	3×1
M	gear reduction ratio vector	$n \times 1$
F	switching surface gain matrix	$n \times n$
W	weighting matrix	$n \times n$
Γ	joint stiffness matrix: diag $\{\gamma_1, \gamma_2, \ldots, \gamma_n\}$, tool path	$n \times n$
K	position gain matrix, spring constant matrix	$n \times n$
L	velocity gain matrix, damping constant matrix	$n \times n$
H	environmental spring constant matrix	6×6
K_P	proportional gain matrix	$n \times n$

TABLE A3-1 *(Continued)*

Symbol	Description	Dimension
K_I	integral gain matrix	$n \times n$
K_D	derivative gain matrix	$n \times n$
ω_n	undamped natural frequency	1×1
ζ	damping ratio	1×1
M_P	peak overshoot	1×1
Δx	pixel spacing in x direction	1×1
Δy	pixel spacing in y direction	1×1
ρ_{xy}	aspect ratio of image: $\Delta x/\Delta y$	1×1
A	area of region R	1×1
x_c	x centroid of region R	1×1
y_c	y centroid of region R	1×1
f	effective focal distance of camera	1×1
μ	coefficient of static friction	1×1
ψ	angular radius of friction cone: atan2 $(\mu, 1)$	1×1
m_{kj}	moment (k, j) of region R	1×1
μ_{kj}	central moment (k, j) of region R	1×1
ν_{kj}	normalized central moment (k, j) of region R	1×1

TABLE A3-2 FUNCTIONS

Symbol	Description	Dimension
$f: U \rightarrow V$	a function f mapping set U into set V	dim (V)
$y^{(k)}(t)$	kth derivative of $y(t)$: $d^k y(t)/dt^k$	1×1
$\delta x(t)$	variation of $x(t)$ about $\hat{x}$: $x(t) - \hat{x}$	$m \times 1$
$V_L(x)$	Liapunov function	1×1
$R_k(\phi)$	kth fundamental rotation matrix	3×3
$R(\phi, u)$	equivalent angle-axis rotation matrix	3×3
Rot (ϕ, k)	kth fundamental homogeneous rotation matrix	4×4
YPR (θ)	composite yaw-pitch-roll transformation matrix	3×3
Tran (p)	homogeneous translation matrix	4×4
Screw (λ, ϕ, k)	kth screw transformation matrix: Tran (λi^k) Rot (ϕ, k)	4×4
Scr (λ, ϕ, u)	general screw transformation along unit vector u	4×4
Scale (c, σ)	homogeneous scaling transformation matrix	4×4
$T_j^k(q)$	coordinate transformation from frame L_k to frame L_j	4×4
$T_{\text{base}}^{\text{tool}}(q)$	arm matrix: $T_0^n(q)$	4×4
$p(q)$	tool-tip position: $H_1 T_0^n(q) i^4$	3×1
$R_j^k(q)$	rotation matrix of frame L_k relative to frame L_j	3×3
$R(q)$	tool orientation: $R_0^n(q) = [r^1(q), r^2(q), r^3(q)]$	3×3
$x^k(q)$	first unit vector of frame L_k: $R_0^k(q) i^1$	3×1
$y^k(q)$	second unit vector of frame L_k: $R_0^k(q) i^2$	3×1
$z^k(q)$	third unit vector of frame L_k: $R_0^k(q) i^3$	3×1
$p^k(q)$	position of origin of frame: L_k: $H_1 T_0^k(q) i^4$	3×1
$r^1(q)$	normal vector: $x^n(q)$	3×1
$r^2(q)$	sliding vector: $y^n(q)$	3×1
$r^3(q)$	approach vector: $z^n(q)$	3×1
$w(q)$	tool-configuration function	6×1

TABLE A3-2 *(Continued)*

Symbol	Description	Dimension
$\bar{w}(q)$	reduced tool-configuration function	$n \times 1$
$w(t)$	tool-configuration trajectory	6×1
$\dot{w}(t)$	tool-configuration velocity: dw/dt	6×1
$D(q)$	manipulator inertia tensor	$n \times n$
$\tilde{D}(q)$	manipulator inertia tensor estimate	$n \times n$
$\Delta D(q)$	manipulator inertia tensor error	$n \times n$
$b(\dot{q})$	joint friction vector	$n \times 1$
$\tilde{b}(\dot{q})$	joint friction vector estimate	$n \times 1$
$\Delta b(\dot{q})$	joint friction vector error	$n \times 1$
$h(q)$	gravity loading vector	$n \times 1$
$\tilde{h}(q)$	gravity loading vector estimate	$n \times 1$
$\Delta h(q)$	gravity loading vector error	$n \times 1$
$C^k(q)$	velocity coupling matrix for joint k	$n \times n$
$c(q, \dot{q})$	velocity coupling vector: $c_k = \dot{q}^T C^k(q)\dot{q}$	$n \times 1$
$\tilde{c}(q, \dot{q})$	velocity coupling vector estimate	$n \times 1$
$\Delta c(q, \dot{q})$	velocity coupling vector error	$n \times 1$
$T(q, \dot{q})$	total kinetic energy of arm: $\dot{q}^T D(q)\dot{q}/2$	1×1
$U(q)$	total potential energy of arm: $-g^T\bar{c}(q)$	1×1
$F(\dot{q})$	generalized force acting on arm: $\tau - b(\dot{q})$	$n \times 1$
$L(q, \dot{q})$	Lagrangian function: $T(q, \dot{q}) - U(q)$	1×1
$D_k(q)$	inertia tensor of link k	3×3
$\bar{c}^k(q)$	position of center of mass of link k	3×1
$\bar{v}^k(q)$	linear velocity of center of mass of link k	3×1
$\bar{\omega}^k(q)$	angular velocity about center of mass of link k	3×1
$v^k(q)$	linear velocity of frame L_k	3×1
$\dot{v}^k(q)$	linear acceleration of frame L_k	3×1
$\omega^k(q)$	angular velocity of frame L_k	3×1
$\dot{\omega}^k(q)$	angular acceleration of frame L_k	3×1
$J^k(q)$	Jacobian matrix of link k	$6 \times n$
$A^k(q)$	linear Jacobian matrix of link k	$3 \times n$
$B^k(q)$	angular Jacobian matrix of link k	$3 \times n$
$J(q)$	manipulator Jacobian matrix	$6 \times n$
$A(q)$	linear manipulator Jacobian matrix: $dp(q)/dq$	$3 \times n$
$B(q)$	angular manipulator Jacobian matrix	$3 \times n$
$V(q)$	tool-configuration Jacobian matrix: $dw(q)/dq$	$6 \times n$
$G(q)$	end-of-arm compliance matrix: $J(q)\Gamma^{-1}J^T(q)$	6×6
$\Delta s^k(q)$	frame displacement of link k: $p^k(q) - p^{k-1}(q)$	3×1
$\Delta r^k(q)$	center-of-mass displacement of link k: $\bar{c}^k(q) - p^k(q)$	3×1
$f^k(q)$	force on link k due to link $k - 1$	3×1
$n^k(q)$	moment on link k due to link $k - 1$	3×1
$s(t)$	speed distribution function	1×1
$\Delta w(\Delta T)$	knot-point deviation	6×1
dex (q)	manipulator dexterity: det $[V^T(q)V(q)]$	1×1
sgn (β)	signum function: sign of β	1×1
$\delta(t)$	unit impulse	1×1
$1(t)$	unit step function	1×1
sat (t)	unit saturation function	1×1
$\sigma(x)$	switching surface function: $\sigma(x) = Fe + w$	$n \times 1$
$X(s)$	Laplace transform of $x(t)$	1×1

TABLE A3-2 *(Continued)*

Symbol	Description	Dimension
$G(s)$	transfer function: $Y(s)/U(s)$	1×1
$G_L(s)$	DC motor and load transfer function: $Q(s)/\tau_L(s)$	1×1
$G_\tau(s)$	torque regulator transfer function: $\tau_L(s)/\tau(s)$	1×1
$G_{\text{PID}}(s)$	PID controller transfer function: $Q(s)/R(s)$	1×1
$i(x, y)$	analog image intensity	1×1
$\nabla i(x, y)$	gradient of analog image	1×2
$I(k, j)$	digital image: $1 \leq k \leq m$, $1 \leq j \leq n$	1×1
$\nabla I(k, j)$	numerical gradient of digital image	1×2
$T_i(k, j)$	ith image template: $1 \leq k \leq m0$, $1 \leq j \leq n0$	1×1
$p(k, j)$	pixel neighborhood function: 0 to 8	1×1
$R(\lambda)$	radius of generalized Voronoi diagram (GVD)	1×1

TABLE A3-3 MATRICES AND VECTORS

Symbol	Description	Dimension
I	identity matrix	$n \times n$
0	zero matrix	$m \times n$
det (A)	determinant of A	1×1
trace (A)	trace of A: $A_{11} + A_{22} + \ldots + A_{nn}$	1×1
$\Delta(\lambda)$	characteristic polynomial of A: det $(\lambda I - A)$	1×1
A^T	transpose of A: $A^T_{kj} = A_{jk}$	$n \times m$
A^{-1}	inverse of A: $A^{-1}A = AA^{-1} = I$	$n \times n$
A^+	Moore-Penrose inverse (pseudoinverse) of A	$n \times m$
$A^{(1)}$	$\{1\}$-inverse of A: $AA^{(1)}A = A$	$n \times m$
$N(A)$	null space of A: $\{x: Ax = 0\}$	nullity (A)
$R(A)$	range space of A: $\{Ax: x \in \mathbf{R}^n\}$	rank (A)
nullity (A)	dimension of $N(A)$	1×1
rank (A)	dimension of $R(A)$	1×1
diag $\{\alpha, \beta, \gamma\}$	diagonal matrix with diagonal elements α, β, γ	3×3
$x \cdot y$	dot product of x and y: $x^T y$	1×1
$x \times y$	cross product of x and y	3×1
$\|x\|$	norm of x: $(x \cdot x)^{1/2}$	1×1
i^k	kth column of identity matrix: $I = [i^1, i^2, \ldots, i^n]$	$n \times 1$
$[p]^X$	homogeneous coordinates of p with respect to X	4×1
p^{base}	base coordinates of p	4×1
p^{camera}	camera coordinates of p	4×1
p^{image}	camera coordinates of image of p	4×1
p^{pixel}	pixel coordinates of image of p	4×1
$p^{\text{aug}}(\lambda)$	augmented camera coordinates of image of p	4×1

TABLE A3-4 COORDINATE FRAMES

Symbol	Description
F	fixed frame: $\{f^1, f^2, f^3\}$
M	mobile frame: $\{m^1, m^2, m^3\}$
C	camera frame: $\{x^c, y^c, z^c\}$
B	robot base frame: $\{x^b, y^b, z^b\}$
T	robot tool frame: $\{x^t, y^t, z^t\}$
P	part frame: $\{x^p, y^p, z^p\}$
L_k	frame assigned to link k: $\{x^k, y^k, z^k\}$
$T_{\text{base}}^{\text{pick}}$	pick position transformation matrix
$T_{\text{base}}^{\text{lift}}$	lift-off position transformation matrix
$T_{\text{base}}^{\text{set}}$	set-down position transformation matrix
$T_{\text{base}}^{\text{place}}$	place position transformation matrix
$T_{\text{image}}^{\text{camera}}$	perspective transformation matrix
$T_{\text{camera}}^{\text{image}}(\lambda)$	inverse perspective transformation matrix
$T_{\text{base}}^{\text{camera}}$	camera calibration matrix
$T_{\text{image}}^{\text{pixel}}$	pixel transformation matrix

TABLE A3-5 SETS

Symbol	Description
$\mathbf{R}^n$	vector space of column vectors with n real components
$\{x \in U: P(x)\}$	set of x in U satisfying property P
$[\alpha, \beta]$	closed interval: $\{x: \alpha \leq x \leq \beta\}$
Q	joint-space work envelope
Υ	work envelope: $\{p(q): q \in Q\}$
Υ_d	dexterous work envelope: $\Upsilon_d \subseteq \Upsilon$
Ω_ρ	domain of attraction of equilibrium point $\hat{x}$
Γ	path in tool-configuration space: $\{w(\lambda): 0 \leq \lambda \leq 1\}$
Γ_m	discrete tool-configuration path: $\{w^k: 0 \leq k < m\}$
Pre (G, v)	preimage of surface G induced by velocity vector v
$B(\hat{x}, \rho)$	open ball: $\{x: \|x - \hat{x}\| < \rho\}$
R_i	region i
B_A	configuration-space obstacle induced by A
S	task-planner source scene
G	task-planner goal scene

TABLE A3-6 OPERATORS

Symbol	Description
$\triangleq$	equals by definition
$\equiv$	is identically equal to
$\in$	is an element of
$\notin$	is not an element of
$\subseteq$	is a subset of
$U \cup V$	union of U and V: $\{x: x \in U \text{ or } x \in V\}$
$U \cap V$	intersection of U and V: $\{x: x \in U \text{ and } x \in V\}$
$\rightarrow$	approaches
lim	limit
min	minimum
max	maximum
exp	exponential
ln	natural logarithm
$O(n^i)$	order of n to the ith power
Re (λ)	real part of λ
$\lvert\beta\rvert$	absolute value of β
Shrink (i)	ith image shrink operator: $0 \leq i \leq 8$
Swell (i)	ith image swell operator: $0 \leq i \leq 8$
Manip (S)	manipulated source scene S

TABLE A3-7 TRIGONOMETRIC SYMBOLS

Symbol	Description
$S\phi$	$\sin \phi$
$C\phi$	$\cos \phi$
$V\phi$	$1 - \cos \phi$
S_k	$\sin q_k$
C_k	$\cos q_k$
S_{kj}	$\sin (q_k + q_j)$
C_{kj}	$\cos (q_k + q_j)$
S_{k-j}	$\sin (q_k - q_j)$
C_{k-j}	$\cos (q_k - q_j)$
atan2 (y, x)	four-quadrant arctan (y/x)

Index

A

Accuracy (*See* Robot specifications)
Active sensing, 307
Actuators, 202
Adaptive control, 289–90
Adept One robot:
  arm matrix, 70–71
  boundary singularity, 189–90
  continuous-path control, 139–40
  horizontal reach, 123
  induced joint torques and forces, 183–84
  kinematic parameters, 70–71, 213
  link coordinates, 68–70, 97, 123, 241
  manipulator Jacobian, 177–79
  photograph, 69
  resolved-motion rate control, 164–66
  soft home position, 71
  straight-line motion, 147–48
  tool-configuration function, 96, 122, 156
  tool-configuration Jacobian, 156–57
  tool deflection, 186–87
  work envelope, 124
Algorithm:
  bounded deviation, 146–47
  composite homogeneous transformation, 45–47
  composite rotation, 36
  Denavit-Hartenberg (D-H) representation, 53–56
  edge detection, 315
  image analysis, 334
  inverse kinematics, Adept One, 99
  inverse kinematics, Intelledex 660, 105
  inverse kinematics, planar articulated robot, 109
  inverse kinematics, Rhino XR-3, 95
  Lagrange-Euler equations, 207–8
  manipulator Jacobian, 177
  polygon penetration, 389
  recursive Newton-Euler equations, 224–25
  region growing, 328
  region labeling, 327
  template matching, 310–11
Android, 3
Applications (*See* Robot applications)
Approach:
  distance, 132
  vector, 53, 71, 89, 126
Arc, 375
Area descriptors, 321–25
Arm equation, 57–61
  arm matrix, 57–60
  example, 60–61
  general, 51, 60, 84
  reduced, 85

Arm matrix, 57–60
  Adept One robot, 70–71
  Intelledex 660 robot, 74–75
  Microbot Alpha II robot, 60–61, 66–67
  planar articulated robot, 106
  Rhino XR-3 robot, 63–66
Aspect ratio, 314
Assembly, robotic, 8, 360
Asymptotic stability (*See* Stability)
Atan2 function, 92
Automation, 1–3
  hard, 1–3
  soft, 2–3
Axes:
  collinear, 159
  major, 4, 9
  minor, 4, 9
  redundant, 9–10

B

Background region, 325–27
Backlash, gear, 68
Ball, open, 246–47
Basis (*See* Coordinate frame)
Binary image, 308, 315, 317
Bin picking problem, 320
Block diagram:
  feedback configuration, 260
  linear system, 257
  parallel configuration, 259
  serial configuration, 259
  summer, 259
Bulk operator, 336–37

C

Camera:
  calibration, 350–53
  coordinates, 109–10
  focus, 111
Capacity (*See* Robot specifications)
Carousels, 128
Centrifugal forces, 206–7
Centripetal forces (*See* Centrifugal forces)
Centroid, 321–22
Chain codes, 319–20
Code:
  chain, 319–20
  gray, 15
  run-length, 318–19
Completeness (*See* Vectors, complete set)
Compliance:
  active, 183, 299, 385–87
  end-of-arm, 184–87
  passive, 387
  pneumatic tool, 4
Compliant motion, 385–87
Computational complexity, 221, 225, 230
Computed-torque control, 283–89
  control law, 284
  error, 284
  three-axis SCARA robot, 286–89
  two-axis articulated robot, 286
Configuration space, 362–68
  convex obstacles, 363–64
  definition, 362
  nonconvex obstacles, 364–65
  rotations, 366–68
  slice projections, 368
  translations, 362–66
Connectedness:
  4-connected region, 328, 335–36
  6-connected region, 336
  8-connected region, 328, 335–36
  set, 321
Constant solution (*See* Equilibrium points)
Constraint, artificial, 302
Constraint, natural, 298, 302
Continuous-path control:
  Adept One robot, 139–40
  Rhino XR-3 robot, 137–39
  (*See also* Rate control)
Continuous-path motion, 8, 135–40
Control:
  law, 265, 277, 284, 295
  speed, 234
  torque, 234
Control problem, 235–36
  regulator, 276
  robot, 235–36
  servo, 283
Control techniques:
  computed-torque, 283–89
  continuous-path rate, 137–40
  impedance, 298–303
  PD-plus-gravity, 276–83
  resolved-motion rate, 164–74
  single-axis PID, 265–76
  variable-structure, 289–98
Convex:
  polygon, 364
  set, 117
Conveyors, 127–28

Coordinate frames, 29–33, 410
Coordinates:
  augmented image, 340–41
  camera, 109–10
  definition, 29
  homogeneous, 41, 339
  image, 337
  link, 51–55
  orthonormal, 30
  pixel, 343–44
Coordinate transformations:
  camera calibration, 350
  composite homogeneous, 45–47
  fundamental homogeneous rotation, 42–43
  fundamental homogeneous scaling, 77–78
  fundamental homogeneous translation, 43–44
  fundamental screw, 49–50, 57–58
  general screw, 50
  homogeneous, 42
  inverse, 32–33
  inverse homogeneous, 47–48
  inverse link, 58
  inverse perspective, 340–44
  link, 58
  orthonormal, 31–32
  perspective, 337–40
Coriolis forces, 206–7
Corner points, 317
Coulomb friction (*See* Friction, static)
Cross:
  correlation, normalized, 312
  product, 28–29
Cubic:
  polynomial paths, 141–42
  splines, 142, 151
Current sensing, 268
Cycle time, 10
Cylindrical-coordinate robot:
  dynamic model, 232
  joint-space work envelope, 148
  kinematic parameters, 113, 232
  link coordinates, 113, 149, 191
  PD-plus-gravity control, 304
  tool-configuration function, 190

D

Damping (*See* Second-order linear systems)
Damping constant, 299
DC motor (*See* Motor, DC servo)
Deflection (*See* Tool, deflection)
Delay time, 263–64
Denavit-Hartenberg (D-H) representation, 53–57
  algorithm, 54
  example, 54–57
Depth measurement (*See* Ranging techniques)
Depth of field, 337
Descriptors, 319–25
  area, 321–25
  line, 319–20
D-H representation (*See* Denavit-Hartenburg (D-H) representation)
Differential motion, 154, 175
Digital-to-analog converter (DAC), 14
Dimension, 27
Direct dynamics (*See* Dynamics)
Direct kinematics (*See* Kinematics, direct)
Displacement vector, infinitesimal, 175, 187
Distance measure, 388
Distribution:
  bimodal, 326
  edge pixel, 316
  speed, 135–40
Domain of attraction, 250, 252–56
Dot product, 26–27
Dual operator, 332
Dynamics, 220–21
  direct, 220
  inverse, 220–21
  symbolic modeling, 225

E

Edge detection, 313–17
  edge fragments, 315–16
  edge pixel density function, 316
  edge pixel distribution function, 316
  edge threshold, 316
  false edges, 315–16
Effective focal distance, 337, 354
Eigenvalues, 247–48
Elongation, 355
Encoder position, 15
End-effector (*See* Tool)
End-of-arm:
  compliance and stiffness, 184–87
  force and moment vector, 182, 203–4
Energy of robotic arm:
  kinetic, 195–200

Energy of robotic arm (*cont.*)
potential, 200–201
Environment, 16, 300–301
Equilibrium points:
impedance control, 301
linear system, 258–59
nonlinear system, 243–44
robot, 244–46
Equivalent:
angle-axis transformation, 38–41
joint torques and forces, 182–90
Error:
steady-state tracking, 261–63
tracking, 265
Estimates, parameter, 280, 283–84
Euler number, 334–35

F

Fault detection:
break fault, 336–37
bridge fault, 336–37
surface defects, 346
Feedback system (*See* Linear feedback systems)
Findspace problem, 394
Fine-motion planning, 381–87
compliant motion, 385–87
friction cone, 382
generalized damper, 381
guarded motion, 382–85
Five-axis articulated robot (*See* Rhino XR-3 robot)
Fixed tools, 128–30
Fixtures (*See* Workspace fixtures)
Focal:
distance, 337, 354
length, 337, 354
Foreground region, 325–27
Forward kinematics (*See* Kinematics, direct)
Four-axis SCARA robot (*See* Adept One robot)
Freeway paths, 368
Friction, 202–4
angle, 382
cone, 382
dynamic, 202–3
gear, 68
static, 202–3, 235
viscous, 202–3
Functions, 407–9
commutative, 27
image, 117
Lagrangian, 195, 201
Liapunov, 251
linear, 27, 41
monotonic, 331–32
piecewise-constant, 347
piecewise-continuous, 347
piecewise-quadratic, 371
positive-definite, 251
variation, 248

G

Gain:
DC, 263
derivative, 265
integral, 265
position, 277
proportional, 265
sliding mode, 292
velocity, 277
Gantry robot (*See* Robot, Cartesian)
Gear reduction, 235–36
Generalized force, 201–4
actuators, 201–2
friction, 202–4
Generalized inverse, 160–63
Generalized Voronoi diagrams (GVD), 368–78
composite, 372–75
edge and vertex, 369–70
graph representation, 375–76
motion heuristics, 375–78
pair of edges, 369
pair of vertices, 370
path validation, 377
Goal:
node, 375
preimage, 383–84
scene, 391–92
surface, 382
Gradient vector:
approximate, 314, 354
exact, 247
Graphics robot simulator (SIMULATR) program, 18
Graphs, 365, 375–76
Grasp planning, 387–80
polygonal parts, 380–81
reachable grasp, 378–79
safe grasp, 378
secure grasp, 379
Gravity, 200–201
Gravity loading vector, 205, 211–12, 218–19, 228, 244
Gray level, 304
Gripper (*See* Tool)
Gross-motion planning:
configuration space, 362–68

generalized Voronoi diagram, 368–78
Guarded motion, 382–85
rotation, 384
translation, 384

H

Hand (*See* Tool)
Height profile, 347
Heuristics, motion, 375–78
Home position:
hard, 16
soft, 56
Homogeneous coordinates, 41–51
composite transformations, 45–48
coordinate frames, 41–42
rotations, 42–43
screw transformations, 49–51
translations, 43–44
Human arm, 160

I

Image:
analysis, 333–34
compression, 318–19
intensity, 311
representation, 308–9
smoothing, 333
texture, 325
Impedance control, 298–303
control law, 299–300
three-axis SCARA robot, 302–3
Indices, 58–59
Induced joint torques and forces (*See* Torques)
Inertia:
common objects, 402–4
moments, 196
principal moment, 196
products, 196
Inertia tensor:
link, 196–98, 209–10, 214–17, 227
manipulator, 199–200, 211, 218, 227
Initial condition, 223–24, 243
Intelledex 660 robot:
arm matrix, 74–75
kinematic parameters, 73–74
link coordinates, 73, 101
photograph, 2, 72
soft home position, 75–76
tool-configuration function, 100
Interpolated motion, 140–45 (*See also* Straight-line motion)
cubic polynomial, 141–42
parabolic blends, 142–45
piecewise linear, 142
Inverse:
generalized, 160–63
perspective transformation, 340–43
{1}-Inverse, matrix:
definition, 171
full-rank computation, 171
linear systems, 172–73
properties, 171–72
Inverse dynamics (*See* Dynamics)
Inverse kinematics (*See* Kinematics, inverse)
Inverted pendulum (*See* One-axis robot)
Iterative image processing, 330–37

J

Jacobian matrix:
general, 247
link, 198–99, 209–11, 214–17, 227
Jacobian matrix, manipulator, 174–82
Adept One robot, 177–79
computation, 176–77
definition, 174–75
partitioned, 175–77
planar articulated robot, 181–82
Rhino XR-3 robot, 179–81
translational motion, 179
Jacobian matrix, tool-configuration:
Adept One robot, 156–57
general, 153–54
planar articulated robot, 157–58
Rhino XR-3 robot, 154–56
Joint:
angle, 51–52, 168–69
coupling, 67–68
displacement, 184
distance, 51–52
prismatic, 4
redundant, 173
revolute, 4
stiffness, 184
type parameter, 58–59
variable, 59
Joint space, 83
Joint-space singularities (*See* Singularities)
Joint-space work envelope (*See* Work envelope)

K

Kinematic parameters:
joint, 51–52

Kinematic parameters (*cont.*)
link, 52–53
Kinematics, direct:
Adept One robot, 68–71
Intelledex 660 robot, 71–76
Microbot Alpha II robot, 60–61
problem, 25–26
Rhino XR-3 robot, 62–66
planar articulated robot, 105–7
Unimation PUMA 200 robot, 78–79
U.S. Robots Maker 110 robot, 79–80
Kinematics, inverse:
Adept One robot, 96–100
elbow up and down solutions, 86, 93, 103
existence of solution, 84–85
Intelledex 660 robot, 100–105
iterative numerical solution, 90
problem, 81–83
Rhino XR-3 robot, 91–96
right- and left-handed solutions, 97, 102, 107
planar articulated robot, 105–9
uniqueness of solution, 85–86
Kinetic energy of robotic arm, 195–200
Knot point, 140, 144, 146–48

L

Laboratory manual, 18
Lagrange:
equation, 195
multiplier, 167
Lagrange-Euler dynamic model (*See* Lagrange-Euler equations)
Lagrange-Euler equations, 204–8
general, 204–8, 235, 301
one-axis robot, 226–28
physical interpretation, 220
three-axis SCARA robot, 212–19
two-axis articulated robot, 208–12
Lagrangian function, 195, 201
Laplace transform, 256–57
Laser, 347–48
Least-squares solutions, 163–64, 172–73
Lens:
fisheye, 361
optical, 337–38
Liapunov function, 251
Liapunov's direct method (*See* Liapunov, second method)
Liapunov's indirect method (*See* Liapunov, first method)
Liapunov stability methods, 247–56
first method, 247–50
second method, 250–56, 277–79
Light, structured, (*See* Structured illumination)
Light patterns, 346–50
Linear:
algebraic systems, 163, 171–73
interpolation with parabolic blends, 142–45
Linear feedback systems, 256–65
block diagram, 257
DC gain, 263
equilibrium points, 258–59
stability, 259
state equation, 258
steady-state tracking error, 261–63
transfer function, 256–57
transient performance, 263–65
Linearization, 248
Line descriptors, 319–20
Link:
inertia tensor, 196–98
Jacobian matrix, 198–99
length, 52–53
twist angle, 52–53
Link coordinates, 51–57
Adept One robot, 68–70
Intelledex 660 robot, 73
Microbot Alpha II robot, 54–56
planar articulated robot, 105–6
Rhino XR-3 robot, 63
List of symbols, 405–11

M

Manipulability, 158
Manipulator:
dexterity, 158
intertia tensor, 199–200
Jacobian, 174–82
(*See also* Robot)
Matrix:
camera calibration, 350–53
composite homogeneous transformation, 45–47
composite rotation, 36
coordinate transformation, 31–32
damping constant, 299
end-of-arm compliance, 185
end-of-arm stiffness, 186
equivalent angle-axis, 38–41
fundamental homogeneous rotation, 42–43
fundamental homogeneous scaling, 77–78

fundamental homogeneous translation, 43–44
fundamental rotation, 33–35
fundamental screw transformation, 49–50
general screw transformation, 50
homogeneous coordinate conversion, 41
identity, 21
inverse homogeneous transformation, 47–48
inverse perspective transformation, 341
Jacobian, 247
joint coupling, 82
joint stiffness, 184
manipulator Jacobian, 174–82
perspective transformation, 339–40
position gain, 277
rotation, 42
sliding mode gain, 292
spring constant, 299
tool-configuration Jacobian, 153–58
velocity coupling, 205
velocity gain, 277
weighting, 167
yaw-pitch-roll transformation, 37–38
zero, 21
Matrix characteristics, 409
block diagonal, 241
diagonal, 209
generalized inverse, 160–63
ill-conditioned, 158
inverse, 21
{1}-inverse, 171–72
multiplication, 36
nullity, 20, 188
null space, 20–21, 187–90
postmultiplication, 37, 45
premultiplication, 37, 45
range space, 20–21, 187–90
rank, 20, 158
singular, 340
skew-symmetric, 278
Microbot Alpha II robot:
arm matrix, 60–61
interior singularity, 159–60
kinematic parameters, 56
link coordinates, 54–56
sketch, 55
soft home position, 66–67
tool-configuration function, 91
Minimum least-squares solution (*See* Pseudoinverse)
Modulation, 346–47
Moments, region:
area, 321–22
central, 322–23
centroid, 321–22
invariant, 325
normalized central, 323–24
principal angle, 324
standard, 321
Moments of inertia, 402–4
Moore-Penrose inverse (*See* Pseudoinverse)
Motion:
compliant, 299, 385–87
continuous-path, 8, 135–40
control methods, 7–8
differential, 154, 175
fine, 133–35
gross, 133–35, 362, 365
guarded, 382–85
interpolated, 140–45
point-to-point, 7, 131–35
self, 159–60, 187–88
straight-line, 83–84, 145–48
Motor, DC servo, 266–68
back emf constant, 267
time constant, 268
torque constant, 267
torque transfer function, 268
Motor, stepper, 266

N

Natural constraints, 298
n-axis planar articulated robot, 168–74
Newton-Euler equations, 221–25
backward equations, 223–24
computational complexity, 228, 230
forward equations, 221–23
initial conditions, 223–24
one-axis robot, 228–30
recursive algorithm, 224–25
Nodes, 375
Noise, 311, 331–33
Nonlinear system:
autonomous, 243
initial condition, 243
nonautonomous, 243
Norm, 27–28
Normal vector, 53
Notation, 18–22, 405–11
coordinate transformations, 21–22, 58–60
list of symbols, 405–11
matrices, 20–21
sets, 19–20
transpose, 19
trigonometric symbols, 22

Notation (*cont.*)
vectors, 19
vector space, 20

O

Obstacle avoidance:
path planning, 362–78
pick-and-place, 133
Occlusion, partial, 320, 393
One-axis robot:
asymptotic stability, 249–50
domain of attraction, 253–56
equilibrium points, 245
kinematic parameters, 226
Lagrange-Euler model, 226–28, 238
link coordinates, 226, 238
manipulator inertia tensor, 227
Newton-Euler model, 228–30
state equations, 237–38
variable-structure control, 296–97
Operating environment (*See* Robot specifications)
Operating modes, 172
Operators, 411
bulk, 336–37
shrink, 330–32
skeleton, 336–37
swell, 332–33
task planner, 391
Optimization:
fixture placement, 360–62
inverse kinematics, 90
weighted joint velocity, 167–68
Orthogonal (*See* Vectors, orthogonal)
Orthonormal (*See* Vectors, orthonormal)
Overshoot, 134, 264–65

P

Parabolic blend, 142–45
acceleration, 143–44
knot-point deviation, 144
trajectory, 145
transition interval, 143
transition time, 143–44
Parallel jaw gripper:
parallel bar design, 379
rack-and-pinion design, 379–80
Parameters, 405–7
joint type, 58–59
kinematic, 51–52
Part:
feeders, 125–27
holding devices, 128–30
ordering, 393–94
presentation, 89–90, 125, 128–29
transport devices, 127–28
Partitioned wrist, 59, 85
Path:
collision-free, 362–78
continuous, 135
cubic polynomial, 141–42
discrete, 140
freeway, 368
piecewise-linear, 142–45
smooth, 140
straight-line, 145–48
validation, 377, 388
Path planning (*See* Gross-motion planning)
PD-gravity control (*See* Proportional, derivative (PD) gravity control)
Peak overshoot, 264
Peg-in-hole problem, 387, 397
Pendulum, inverted (*See* One-axis robot)
Perspective:
transformation, 337–40
vector, 42, 339
Phase-locked loop, 303
Pick-and-place operation, 131–35
lift-off point, 131–32
pick point, 131–32
place distance constraint, 132
place point, 132
set-down point, 133
speed variation, 134–35
via points, 133–34
PID control (*See* Proportional, integral, derivative (PID) control)
Pitch:
global tool, 92, 119
screw, 49
tool, 11–12, 87
Pixel:
coordinates, 343–44
definition, 308
function, 330–31
interior, 331
Planar articulated robot:
arm matrix, 106
interior singularity, 160
{1}-inverse rate control, 173–74
kinematic parameters, 106, 208
link coordinates, 106, 192, 239
manipulator Jacobian, 181–82
pseudoinverse rate control, 169–70
soft home position, 106

tool-configuration function, 107, 157, 169
tool-configuration Jacobian, 157–58, 169
Planning:
fine motion, 381–87
grasp, 378–81
gross motion, 368–78
planar simulation, 388–90
task, 391–96
Point-to-point motion, 7
Poles of linear system, 259
Polygonal penetration, 389–90
Polyhedral objects, 313–19
Postmultiplication, 37, 45
Potential energy of robotic arm, 200–201
Precision:
camera, 361
robot, 13–15, 68, 360–61
Preimage of goal, 383–84
Premultiplication, 37, 45
Principal:
angle, 324
axis, 192, 324
Prismatic joint, 4
Problems, 22–23, 76–80, 112–15, 148–51, 190–93, 231–32, 303–6, 353–55, 396–98
Product:
cross, 28–29
dot, 26–27
Productivity, 135
Proportional, derivative (PD) gravity control, 276–83
control law, 277
three-axis SCARA robot, 281–82
two-axis articulated robot, 281
stability, 277–79
Proportional, integral, derivative (PID) control, 265–76
stability, 273–74
transfer function, 272–73
three-axis SCARA robot, 275–76
Pseudoinverse, 160–63
definition, 160–61
full-rank computation, 161–62
linear systems, 163
Pulse-width-modulated (PWM) signal, 268–69
PWM signal (*See* Pulse-width-modulated signal)

Q

Quantization, 308

R

Ranging techniques, 340, 347–50
Rate control:
Adept One robot, 164–66
redundant robots, 166–70, 173
resolved-motion, 164–70
(*See also* Continuous-path control)
Reach (*See* Robot specifications)
Reach constraint, 120–21
Recursive Newton-Euler formulation (*See* Newton-Euler equations)
Redundant planar articulated robot:
rate control with {1}-inverse, 172–74
resolved-motion rate control, 169–70
tool-configuration function, 169
tool-configuration Jacobian, 169
Redundant robots, 85–86, 166–74
Reference input, 265
Region:
4-connected, 328
8-connected, 328
definition, 321
growing, 326–29
labeling, 326–30
Region painting (*See* Region, growing)
Regulator:
speed, 304–5
torque, 268–72
Remote Center Compliance, 387
Repeatability (*See* Robot specifications)
Residual vector, 163
Resolved-motion rate control (*See* Rate control)
Revolute joint, 4
Revolute robot (*See* Robot, articulated)
Rhino XR-3 robot:
arm matrix, 63–66
continuous-path control, 137–39
joint coupling, 67–68
kinematic parameters, 63–64
link coordinates, 63, 91, 118
manipulator Jacobian, 179–81
photograph, 17, 62
reach constraint, 120–21
soft home position, 65
specifications, 16–18
tool-configuration function, 91, 119, 155
tool-configuration Jacobian, 154–56
work envelope, 118–22
Ringing, 135
Robert's cross operator, 315
Robot:
anthropomorphic, 7
articulated, 7

Robot (*cont.*)
Cartesian, 5
clean room, 16
commercial, 235–36
cylindrical, 5, 11,113
definition, 3
direct-drive, 68, 236
general, 9, 85
kinematically redundant, 85–86, 166–70
kinematically simple, 52, 70
planar articulated, 105–9, 168–70
SCARA, 6
single-axis, 30–32, 76, 226–30
spherical, 6, 114
Robot applications, 8–9
Robot classification, 3–8
drive technologies, 3–4
motion control methods, 7–8
work-envelope geometries, 4–7
Robot market, 8
Robot population, 9
Robot programming:
languages, 359
task-level, 359
teach method, 358
Robot specifications, 9–18
accuracy, 15–16
load-carrying capacity, 10
number of axes, 9
operating environment, 16, 71
precision, 13–15, 68, 360–61
reach and stroke, 10–11, 120–21
repeatability, 13, 71
Rhino XR-3 robot, 16–18
speed, 10
tool orientation, 11–12
Robust control, 289, 298
Roll:
global tool, 99
shoulder, 101–2
tool, 11–12, 87–88
Rotations, 33–41
composite, 36–41
equivalent angle-axis, 38–41
fundamental, 33–35
fundamental homogeneous, 42–43
infinitesimal, 174
inverse, 44
pure, 49
yaw-pitch-roll transformation, 37–38
Routh-Hurwitz stability test, 273–74
Row operations, 93
Run-length encoding, 318–19

S

Salt-and-pepper noise, 333
Scale:
factor, 41–42, 339
function, 87–88
transformation, 77–78
SCARA robot (*See* Adept One robot)
Scene analysis, 393
Scenes, 391–92
Screw transformations, 49–51
Second-order linear systems:
critically damped, 264, 285
damping ratio, 263
DC gain, 263
overdamped, 264–65
undamped natural frequency, 263
underdamped, 264
Seed (*See* Region, growing)
Segmentation of image, 325–30
gray scale histogram, 325–26
texture, 325
thresholding, 325
Selective compliance assembly robot arm (*See* Adept One robot)
Self-motions, 159–60, 187–88
Servo, 283
Set point, 265
Sets, 410
Settling time, 264
Shape analysis, 319–25, 330
Shortest path, 365–66
Shrink operator, 330–32
Silhouettes, 345
Simulation:
computed-torque control, 286–89
force sensor, 390
graphics robot, 18
PD-plus-gravity control, 281–82
planar motion, 388–90
task planner, 394–96
SIMULATR (*See* Graphics robot simulator program)
Single-axis PID control (*See* PID control)
Singular configuration (*See* Singularities)
Singularities:
boundary, 158–59
induced torques, 188–90
interior, 112, 159–60
joint-space, 158–60, 187–90
self-motions, 187–88
tool displacements, 187–88
workspace, 158

Six-axis articulated robot (*See* Intelledex 660 robot)
Skeleton operator, 336–37
Sliding:
  mode, 291–93
  vector, 53
Source:
  node, 375
  scene, 391–92
Spatial resolution, 308
Specifications (*See* Robot specifications)
Speed:
  arm, 10, 235
  distribution function, 135–40
  profile, 136
  regulator, 304–5
  variation, 134–35
Spherical-coordinate robot:
  joint-space work envelope, 149
  kinematic parameters, 114
  link coordinates, 114, 149, 192
  PD-plus-gravity control, 304
  tool-configuration function, 192
Spherical wrist (*See* Wrist, spherical)
Splines, cubic, 142, 151
Spring constant, 184, 299
Stability:
  asymptotic, 246–47
  Liapunov's first method, 247–50
  Liapunov's second method, 250–56
  linear system, 259
  one-axis robot, 249–50
  PID controller, 273–74
  Routh-Hurwitz test, 273–74
Stack data structure, 327–29
Stacking operations, 132–33
State, 236–37, 243
State equations, 236–37
  constant solutions, 243–56
  one-axis robot, 237–38
  three-axis SCARA robot, 240–43
  two-axis articulated robot, 238–40
Steady-state tracking, 261–63
Stepper motor, 266
Stiffness:
  end-of-arm, 185–86
  joint, 184
Straight-line motion, 145–48
  approximate, 146
  bounded-deviation, 146–48
  uniform, 145
Stroke (*See* Robot specifications)
Structure, manipulator, 184, 188–90
Structured illumination, 345–50
  back lighting, 345
  front lighting, 345–46
  light patterns, 346–50
  side lighting, 346
  triangulation, 348–50
Structured light (*See* Structured illumination)
Subscripts, 19–20, 58–60
Summer, 259
Superscripts, 19–20, 58–60
Surface defects, 346
Swell operator, 332–33
Switching:
  surface, 291
  time, 292–93
Symbols, special, 405–11

T

Task-level programming, 358–59
Task planner, 357, 391–96
  fine-motion planning, 381–87
  grasp planning, 378–80
  gross-motion planning, 362–78
  part ordering, 393–94
  scene analysis, 393
  simulation, 394–96
  subproblems, 393
Template matching, 309–13
  corner points, 317
  performance index, 310, 312
  template, 309
Tessellation:
  configuration space, 368
  image, 336
Test pattern, 351
Threading operation, 129–30, 140
Three-axis planar articulated robot (*See* Planar articulated robot)
Three-axis SCARA robot:
  computed-torque control, 286–89
  dynamic model, 212–19
  manipulator inertia tensor, 218
  PD-plus-gravity control, 281–82
  PID control, 275–76
  state equations, 240–43
Thresholding of image, 325–26
Tool:
  configuration, 82
  definition, 1
  deflection, 185–87
  displacement, 187–88
  fixed, 128–30
  length, 85

Tool (*cont.*)
orientation, 11–13, 53
parallel-jaw gripper, 379–80
path, 135, 142
trajectory, 135–36
Tool configuration function, 90
Adept One robot, 96
Intelledex 660 robot, 100
planar articulated robot, 107
Rhino XR-3 robot, 91
Tool-configuration Jacobian (*See* Jacobian matrix, tool-configuration)
Tool-configuration space, 83
Tool-configuration vector, 87–90
Adept One robot, 89–90
definition, 87–88
reduced, 95, 100
Rhino XR-3 robot, 88–89
Torque regulator, 268–72
maximum torque, 271
saturation, 271–72, 289
transfer function, 270–71
Torques, induced joint, 182–90
Adept One robot, 183–84
definition, 182–83
Tracking error, 265
Trajectory, tool, 135–37
Transfer function, 256–61
closed-loop, 260–61
DC motor, 268
definition, 256
negative feedback configuration, 260
open-loop, 260
parallel configuration, 259
PID controller, 272–73
serial configuration, 259
speed regulator, 304–5
torque regulator, 270–71
Transformation (*See* Coordinate transformation)
Transient performance, 263–65
damping ratio, 263
delay time, 263–64
peak overshoot, 264
settling time, 264
undamped natural frequency, 263
Translation:
homogeneous matrix, 43–44
infinitesimal, 174
inverse homogeneous matrix, 44
pure, 49
vector, 42
Triangulation, 348–50
Trigonometric:
identities, 400–401
symbols, 411
Tuning, parameter, 265
Two-axis articulated robot:
computed-torque control, 286
dynamic model, 208–12
equilibrium points, 245–46
manipulator inertia tensor, 211
PD-plus-gravity control, 281
state equations, 238–40

U

Uncertainty, 359–62
Unimation PUMA 200 robot, 78
U.S. Robots Maker 110 robot, 79

V

Variable-structure control, 289–98
control law, 295
sliding mode, 291–93
switching surface, 291
Variation, function, 248
Vectors:
approach, 53, 71
complete set, 27
force and moment, 182
gradient, 247
gravity loading, 205
joint limit, 82
normal, 53
orthogonal, 27
orthonormal, 28, 84
perspective, 42
residual, 163
sliding, 53
tool configuration, 87–88
translation, 42
unit, 27–28
velocity coupling, 206
Velocity coupling:
matrix, 205, 211–12, 218–19, 228
vector, 206–7, 235
Vibratory bowl part feeder, 125
Virtual work, 182, 202–3

W

Work cell, robotic, 109–11
Work envelope:
Adept One robot, 122–24
dexterous, 117
geometries, 3–7

B75017
ΛΡΙΑΚΟ
ΒΝ.5707

joint-space, 82, 117, 119, 122, 148–49
Rhino XR-3 robot, 118–22
total, 117
Workspace analysis, 116–18
Workspace fixtures, 124–30, 360
carousel, 128
conveyor, 127–28
fixed tool, 127–30
part feeder, 125–27
vibratory bowl, 125
World model, 359
Wrist, spherical:
analysis, 85
definition, 12
example, 56–57

Y

Yaw, 11–12, 72, 87
Yaw-pitch-roll (YPR) transformation, 37–38, 114

FUNDAMENTALS OF ROBOTICS
Analysis & Control

by

Robert J. Schilling
Associate Professor, Clarkson University

This lucid text is a comprehensive introduction to the fundamentals of robotics and to the analysis and control of industrial robots. The book, designed for senior-level undergraduates and beginning-level graduate students in the fields of engineering and computer science, will also serve as a reference for practicing engineers. The topics are developed in nine comprehensive chapters. The first four chapters lay down the basic foundations in robotic manipulation, the algebraic "arm equation" and its solution, and techniques for planning robot motions. The remaining chapters cover more advanced topics including differential motion and statics, manipulation dynamics, torque-based control techniques, robotic vision, and high-level task planning.

The following prominent features promote the understanding of the dynamic field of robotics:

- Case study type examples of educational, industrial, and generic robots are used throughout the text including: five-axis Rhino XR-3, four-axis Adept One, six-axis Intelledex 660, and three-axis planar manipulator.
- Complete kinematic solutions for several important generic classes of robotic arms.
- Numerous examples, exercises, and problems in each chapter.

ISBN 81-203-1047-0
9 788120 310476

To learn more about
Prentice-Hall of India products,
please visit us at : www.phindia.com

Rs. 195.00

EEE

Schilling

FUNDAMENTALS OF ROBOTICS
Analysis & Control

PHI
Prentice-Hall